Exploring advanced forecasting methods with applications in aviation

by

EVANS MOGOLO RIBA

submitted in accordance with the requirements for the degree of

MASTER OF SCIENCE

in the subject

OPERATIONS RESEARCH

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: PROF MD JANKOWITZ

February 2021

Declaration of Authorship

I, EVANS MOGOLO RIBA, declare that this thesis titled, "Exploring advanced forecasting methods with applications in aviation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

More time series forecasting methods were researched and made available in recent years. This is mainly due to the emergence of machine learning methods which also found applicability in time series forecasting. The emergence of a variety of methods and their variants presents a challenge when choosing appropriate forecasting methods.

This study explored the performance of four advanced forecasting methods: autoregressive integrated moving averages (ARIMA); artificial neural networks (ANN); support vector machines (SVM) and regression models with ARIMA errors. To improve their performance, bagging was also applied. The performance of the different methods was illustrated using South African air passenger data collected for planning purposes by the Airports Company South Africa (ACSA). The dissertation discussed the different forecasting methods at length. Characteristics such as strengths and weaknesses and the applicability of the methods were explored. Some of the most popular forecast accuracy measures were discussed in order to understand how they could be used in the performance evaluation of the methods.

It was found that the regression model with ARIMA errors outperformed all the other methods, followed by the ARIMA model. These findings are in line with the general findings in the literature. The ANN method is prone to overfitting and this was evident from the results of the training and the test data sets. The bagged models showed mixed results with marginal improvement on some of the methods for some performance measures.

It could be concluded that the traditional statistical forecasting methods (ARIMA and the regression model with ARIMA errors) performed better than the machine learning methods (ANN and SVM) on this data set, based on the measures of accuracy used. This calls for more research regarding the applicability of the machine learning methods to time series forecasting which will assist in understanding and improving their performance against the traditional statistical methods.

Keywords

Time series forecasting; regression model with ARIMA errors; autoregressive integrated moving averages; ARIMA; artificial neural networks; ANN; support vector machines; SVM; bagging; bootstrap aggregating; air passengers

Acknowledgements

I would like to thank the Lord Almighty for the strength, health and wisdom to help me navigate the whole process of writing the dissertation. Special gratitude also go to the following people:

- Professor M D Jankowitz for her dedication and great support she showed me as my supervisor for the project. She was always available to answer any question. When I had trouble accessing the data for the study, she came through for me with suggestions and possible way forward. I am humbled for the in depth and thorough review of my work and recommendations she made from the proposal writing through to the end of the dissertation.
- Ms Alexa Barnby who took her time out and assisted in the editing of the document and ensuring that the language used is consistent and of acceptable standards.
- My family, for the support they gave and ensuring that my writing sessions were as comfortable as possible. The patience that they had even when it was clearer that I had sacrificed the time I should have spent with them in favour of this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Forecasting is an essential planning tool for organisations, enabling them to better understand what to expect in the near future. Certain environmental factors have an impact on businesses that operate in that particular environment. Hence, the ability of businesses to understand and be able to predict some of these factors and associated events enables them to plan better.

One of the most critical steps in the forecasting process is choosing and fitting models. There are a number of forecasting models to choose from. These models can be grouped into two basic categories, namely qualitative and quantitative methods. Qualitative methods rely on human judgement, the so-called expert knowledge or intuition, and become more valuable when there is no historical data available. Some of the most popular qualitative forecasting methods include Delphi, market surveys and expert opinion.

In contrast to qualitative methods, quantitative forecasting methods rely on historical data where mathematical models are fitted based on such data. These methods are further grouped into two subcategories, namely causal and time series forecasting methods. Time series methods assume that the historical data is made up of four components, namely trend, seasonality, cycle and an irregular component. These components are extracted from the data and form the basis for forecasting.

In recent years, there has been an increase in the number of time series forecasting methods available to the forecaster. This is mainly due to the emergence of the machine learning methods which also found applicability in time series forecasting. These methods include the artificial neural networks and support vector machines. The increase in the number of time series methods has also presented some challenges when deciding on the forecasting methods to consider for evaluation. In order to decide on a quantitative forecasting model to investigate, one needs to understand certain properties of the different models that can be observed from the historical data. As it is impossible to investigate all the available models, knowing some of these properties will enable the forecaster to narrow down the list of methods to consider and save time in the process.

The literature shows wide applicability for the advanced forecasting methods. These include Widowati et al. (2016), Ramos, Santos, and Rebelo (2015), Sánchez Lasheras et al. (2015), Taneja et al. (2016), Yuan, Liu, and Fang (2016), Oliveira, Steffen, and Cheung (2017), Gibrilla, Anornu, and Adomako (2018), and Hikichi, Salgado, and Beijo (2017) for autoregressive integrated moving averages, Kaur, Kumar, and Segal (2016), Panapakidis and Dagoumas (2016), and Keles et al. (2016) for artificial neural networks, Francis, Tay, and Cao (2001), Kim (2003), and Cao and Tay (2001) for support vector machines and Van den Bossche, Wets, and Brijs (2004) for regression models with ARIMA errors. A comparison of advanced forecasting methods can be found in Ahmed et al. (2010) and Makridakis, Spiliotis, and Assimakopoulos (2018).

The literature on the comparison of these methods is mainly quantitative and based on quantitative measures of accuracy. The application of these models to the South African aviation data is also limited. This study is aimed at studying and applying advanced forecasting methods to South African air passenger data. Different advanced forecasting models will be explored and used to forecast the number of air passengers passing through airports. Accuracy measures will also be used to compare the performance of different methods and determine which model produces the most suitable forecasts.

The objectives of this study are to:

- explore and compare advanced forecasting methods
- illustrate the performance of the methods using South African air passengers data
- evaluate and select the most suitable method for forecasting South African air passenger data.

This dissertation consists of five chapters. Chapter 1 introduces the background and aims of the study as well as giving a summary of the document. In Chapter 2, the different forecasting methods are introduced, the literature on these methods is reviewed and the advantages and disadvantages of the various methods are discussed. Chapter 3 reviews the most important measures of forecasting accuracy. The performance of the different methods is evaluated in Chapter 4. In this chapter, the monthly air passenger data set used is introduced and the models are also bagged to improve their performance. Finally, conclusions and recommendations are made in Chapter 5.

Chapter 2

Advanced Forecasting Methods

2.1 Introduction

The literature presents a number of forecasting methods that can be used, but most of them work well only under certain conditions. In order for a method to work, the data need to behave in a certain way. Some methods work well when the data have no trend, others when the variance in the data is constant. When the data have no trend over time, they are said to be stationary in the mean. This means that the time series has a constant mean. Differencing the time series that is not stationary in the mean will change the time series to one that is stationary in the mean. When the time series shows no changes in variance over time, it is said to be stationary in the variance. In most cases, transforming the data using some mathematical function will stabilise the variance.

In recent years a number of more advanced forecasting methods have begun to gain popularity (Aladag and Eğrioğlu, 2012). Despite this, the autoregressive integrated moving averages (ARIMA) model remains one of the most widely used forecasting methods across different industries (Adhikari and Agrawal, 2013). These models were made popular by Box and Jenkins during the 1970s (Hyndman and Athanasopoulos, 2018).

Some of the advanced forecasting methods include the artificial neural network (ANN),

the regression model with ARIMA errors, the support vector machine (SVM) and bootstrapping and bagging. These methods have proved to be superior to some of the traditional models for certain types of data. They are also able to handle nonlinear time series data and as a result are used mainly to forecast financial time series data (Sapankevych and Sankar, 2009).

In this chapter, advanced forecasting methods are discussed. Section 2.2 to Section 2.5 discuss the different forecasting methods, followed by a discussion on bagging in Section 2.6. The final sections examine the advantages and disadvantages of the different methods and the implementation of these methods in R, a language that provides a variety of statistical and graphical techniques.

2.2 ARIMA Models

ARIMA models are some of the most popular methods used to analyse time series data (Adhikari and Agrawal, 2013). Time series models assume that everything that needs to be known about the possible state of the system in the future depends solely on the behaviour of that system in the past and the present. Using past observations, a mathematical model is then created, which is subsequently used to predict future outcomes. This is called time series forecasting.

ARIMA models are being used for forecasting across several industries. Gibrilla, Anornu, and Adomako (2018) used an ARIMA model to forecast the groundwater levels in the White Volta River basin of Ghana, while Hikichi, Salgado, and Beijo (2017) used an ARIMA model to forecast the number of ISO 14001 certifications in the Americas. The use of ARIMA models can also be observed in other areas (Widowati et al., 2016; Ramos, Santos, and Rebelo, 2015; Sánchez Lasheras et al., 2015; Taneja et al., 2016; Yuan, Liu, and Fang, 2016; Oliveira, Steffen, and Cheung, 2017).

Like most univariate time series methods, an ARIMA model assumes that all the information needed to predict the future is contained in the history of the time series. The

general ARIMA model is represented by equation 2.1.

$$(1 - \phi_1\mathbf{B} - \phi_2\mathbf{B}^2 - \ldots - \phi_p\mathbf{B}^p)(1 - \mathbf{B})^d Y_t = \vartheta_0 + (1 - \theta_1\mathbf{B} - \theta_2\mathbf{B}^2 - \ldots - \theta_q\mathbf{B}^q)\varepsilon_t. \quad (2.1)$$

The variables Y_t and ε_t are the actual value and the error term of the time series at time t respectively. The coefficients ϕ_i and θ_i where $i = 0, 1, 2, \ldots$ are model parameters for the autoregressive (AR) and moving average (MA) to be estimated respectively. The variables p and q represent the order of the AR and the MA part of the model respectively and ϑ_0 is a constant term. The errors are assumed to be normally distributed with a mean of 0 and a variance of σ^2. The $\mathbf{B}$ is called the shift operator and is defined as $\mathbf{B}^s Y_t = Y_{t-s}$.

The ARIMA model in 2.1 can be represented by the notation ARIMA(p, d, q). The d here represents the order of differencing and corresponds to the I in ARIMA. Varieties of this model exist, in fact when d and q equal zero the model can be represented as AR(p), an autoregressive model of order p.

2.2.1 Dealing with Non-stationarity in Time Series

A time series is said to be stationary if the properties of the series do not change over time. Therefore, a time series that shows signs of trend or seasonality is not stationary. A plot of the time series could be used to check whether the series is stationary, for example if the plot shows no sign of changes in the mean of the data, the series is said to be stationary in the mean. By the same token, if the time series does not show any signs of change in variation of the observations over time, the series is said to be stationary in the variance. Non-stationarity in the variance can be removed by transforming the series using a mathematical function such as log or square root depending on the pattern.

When the time series is not stationary in the mean, differencing the time series has the effect of transforming it to one that is stationary. The first difference Y'_t is defined as

$$\begin{aligned} Y'_t &= Y_t - Y_{t-1} \\ &= (1 - \mathbf{B})Y_t, \end{aligned} \tag{2.2}$$

where Y_t is the observation of the series at time period t and $\mathbf{B}$ is the backwards shift operator. The second difference Y''_t is defined as

$$\begin{aligned} Y''_t &= Y'_t - Y'_{t-1} \\ &= (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \\ &= Y_t - 2Y_{t-1} + Y_{t-2} \\ &= (1 - 2\mathbf{B} + \mathbf{B}^2)Y_t \\ &= (1 - \mathbf{B})^2 Y_t. \end{aligned} \tag{2.3}$$

Figure 2.1 presents a time plot of a simulated non-stationary time series. This time series was differenced once and the results are shown in Figure 2.2.

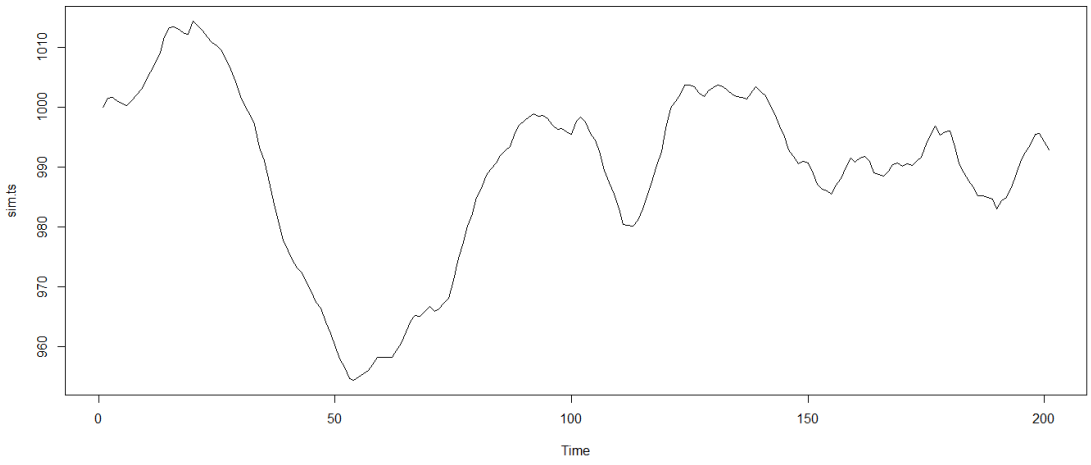


Figure 2.1: Simulated non-stationary time series

The plot in Figure 2.2 suggests that the data are stationary. The data fluctuate around a

constant mean and there appears to be no evidence of change in the variance of the data over time. It is important to identify non-stationarity in a time series as this enables the fitting of a proper ARIMA model. If the time series is not stationary in the mean, one needs to take the first difference of the series. If, after differencing, the series still shows signs of non-stationarity, then further differencing can be done until the series becomes stationary. This will be represented by the value of d in the ARIMA(p, d, q) model.

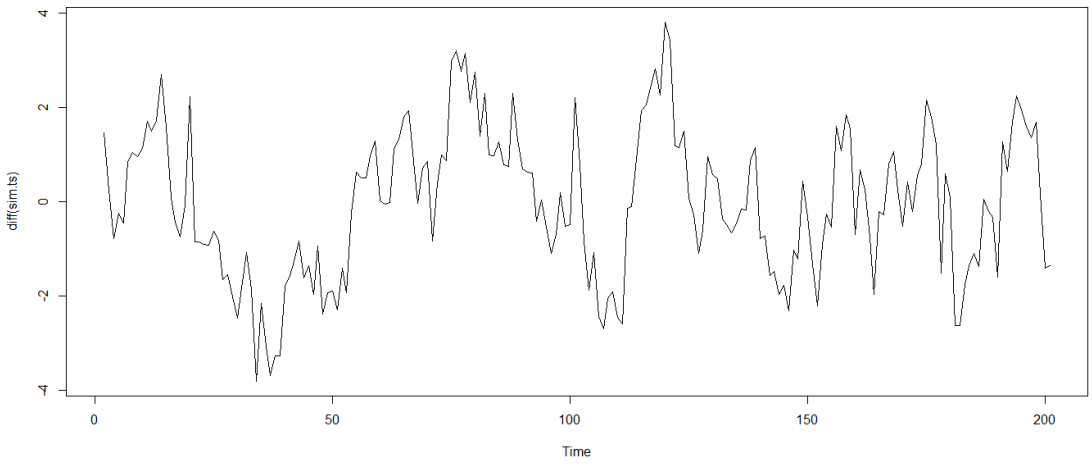


Figure 2.2: First difference of simulated non-stationary time series

The other form of differencing is called seasonal differencing. This is the difference between observations that are a season apart. For seasonal data, the seasonal differencing Y'_t can be defined as

$$\begin{aligned} Y'_t &= Y_t - Y_{t-s} \\ &= (1 - \mathbf{B}^s)Y_t, \end{aligned} \tag{2.4}$$

where s is the length of the season, for example $s = 4$ for quarterly data and $s = 12$ for monthly data.

For a seasonal time series, a seasonal difference is recommended. If the time series remains non-stationary after seasonal differencing of the data, first differencing can be applied to the resulting data. It should be noted that when both the seasonal and the

first differences are applied, it does not matter which difference is applied first, as the results will not be affected. Sometimes taking the seasonal difference first will result in a stationary series and there will consequently be no need for further differencing (Hyndman and Athanasopoulos, 2018).

2.2.2 Autoregressive and Moving Average Model

An AR(p) model is a special ARIMA(p, d, q) model where $d = 0$ and $q = 0$. The model can also be written as ARIMA($p, 0, 0$). An AR(p) model is defined as

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + e_t, \tag{2.5}$$

where c is a constant term, ϕ_i is the i^{th} autoregressive parameter and e_t is the error term at time t.

The other form of the ARIMA model occurs when $p = 0$ and $d = 0$. This form is represented as MA(q) or ARIMA($0, 0, q$), a moving average model of order q. The model is defined as

$$Y_t = c + \varepsilon - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q}, \tag{2.6}$$

where c is a constant term, θ_i is the i^{th} moving average parameter and ε_{t-q} is the error term at time $t - q$.

2.2.3 The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

In time series analysis, the autocorrelation coefficient of a series is defined as

$$r_k = \frac{\sum_{t=k+1}^{n} (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n} (Y_t - \bar{Y})^2}, \tag{2.7}$$

where $\bar{Y}$ is the mean of the time series data and $k = 1, 2, \ldots$ is the lag in the series. r_k is called the autocorrelation coefficient of the time series at lag k. For example r_1 shows

how successive observations relate to each other and r_2 shows how observations two periods apart relate to each other. In general r_k shows how values k periods apart relate to each other. The autocorrelation coefficients at lags $1, 2, 3, \ldots$ together form what is called the autocorrelation function (ACF). The plotted ACF makes it easier to understand the result. The ACF is one of the most useful tools used to identify an ARIMA model.

Another tool used for the purpose of identifying the relevant ARIMA model is the partial autocorrelation Function (PACF). The PACF is formed by stringing the partial autocorrelations of a series together. A partial autocorrelation is a conditional correlation between y_t and y_{t-p} given the observations that lie between y_t and y_{t-p}, namely $y_{t-p+1}, \ldots, y_{t-1}$. By definition, the first order partial autocorrelation coefficient is the same as the first order autocorrelation (Eberly College of Science, 2019). Partial autocorrelation coefficients are used to measure the degree of association between Y_t and Y_{t-p} when the effects of other time lags (1, 2, 3, $\ldots, k-1$) are removed (Hyndman and Athanasopoulos, 2018). The partial autocorrelation coefficient α_p is calculated by regressing Y_t against $Y_{t-1}, \ldots, Y_{t-p}$ as

$$Y_t = b_0 + b_1 Y_{t-1} + b_2 Y_{t-2} + \ldots + b_p Y_{t-p}. \tag{2.8}$$

The partial autocorrelation coefficient α_p is estimated by the b_p in equation 2.8. Figure 2.3 displays an ACF and a PACF of a simulated time series.

The dashed horizontal lines on the ACF and PACF plots show the critical values for the lags r_i. For a white noise model, the majority of the r_is should fall within these lines. According to Quenouille (1949), a white noise model has approximately independent and identically distributed autocorrelations with standard error $\frac{1}{\sqrt{n}}$. This is the reason why the critical values are drawn at $\pm\frac{1.96}{\sqrt{n}}$, where n is the number of observations. The ± 1.96 ensures that 95% of the lags lie between these limits for white noise series.

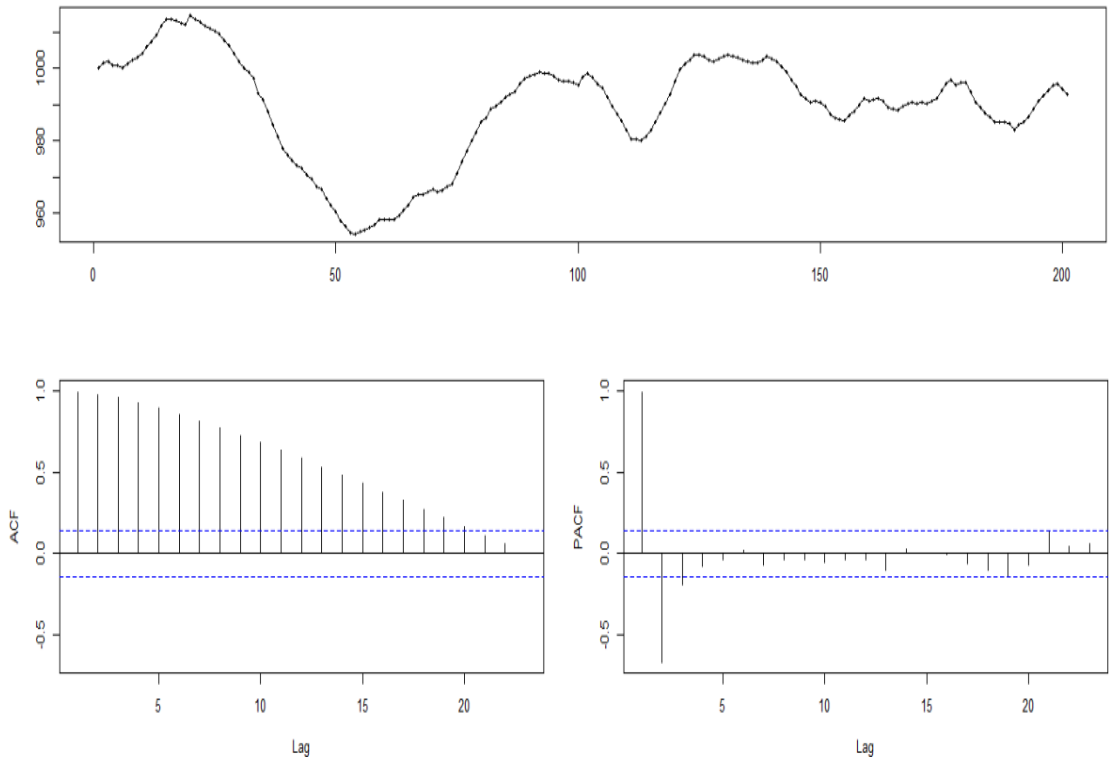


Figure 2.3: ACF and PACF of a simulated non-stationary time series

2.2.4 Seasonal Considerations in ARIMA Models

Seasonality in a time series can be identified by observing the behaviour of the auto-correlations at seasonal lags. For example a monthly time series has seasonal lags at $r = 12, 24, 36, \ldots$ and a quarterly series would have seasonal lags at $r = 4, 8, 12, \ldots$. If any seasonality exists, the seasonal lags in the ACF or PACF will show large spikes that are significantly different from zero. An ARIMA model with a seasonal component is written as ARIMA$(p, d, q)\,(P, D, Q)_s$ where P, D and Q are the seasonal orders of the AR, seasonal differencing and MA respectively. An ARIMA model of this form is also referred to as the seasonal ARIMA model or SARIMA for short. Using the backward shift operator a SARIMA model can be presented as

$$\phi(\mathbf{B})\mathbf{\Phi}(\mathbf{B}^s)(1-\mathbf{B})^d(1-\mathbf{B}^s)^D y_t = \delta + \theta(\mathbf{B})\mathbf{\Theta}(\mathbf{B}^s)\eta_t, \tag{2.9}$$

where δ is the constant term, and the error term at time t is represented by η_t. Further more

$$\phi(\mathbf{B}) = 1 - \phi_1\mathbf{B} - \ldots - \phi_p\mathbf{B}^p \tag{2.10}$$

$$\mathbf{\Phi}(\mathbf{B}) = 1 - \mathbf{\Phi}_1\mathbf{B} - \ldots - \mathbf{\Phi}_P\mathbf{B}^P \tag{2.11}$$

$$\theta(\mathbf{B}) = 1 - \theta_1\mathbf{B} - \ldots - \theta_q\mathbf{B}^q \tag{2.12}$$

$$\mathbf{\Theta}(\mathbf{B}) = 1 - \mathbf{\Theta}_1\mathbf{B} - \ldots - \mathbf{\Theta}_Q\mathbf{B}^Q. \tag{2.13}$$

2.2.5 ARIMA Model Identification

Since there are many ARIMA models, the ACF and PACF become increasingly instrumental when identifying a suitable ARIMA model. The first step in model identification is to fit a tentative model. The following steps show how to go about choosing a tentative model:

- Plot the time series and try to identify any unusual observations and non-stationarity in the variance of the series over time. The unusual observations need to be dealt with and transformations made to the data to achieve stationarity in the variance.
- The next step is to plot the ACF, the PACF and the time plot of the time series. Check whether the data show any signs of trend or seasonality. If the ACF or the PACF drops rapidly to zero or the time plot shows a horizontal line with a constant mean, then the series is stationary.
- If the series does not appear to be stationary then the data should be differenced. For non-seasonal data, the first difference should be taken, otherwise seasonal differencing should be applied to the data. Again plot the ACF, the PACF and the time plot of the differenced data to determine whether the series is stationary. If the series still appears to be non-stationary, take the difference of the differenced series. In most cases, this should be enough to obtain stationarity.

- Plot the ACF and the PACF to examine any patterns in the time series. The pattern of the autocorrelations will reveal one of the following:
  - Signs of seasonality: these can be identified by large ACF or PACF values which are significantly different from zero at seasonal lags. For monthly data the ACF will be large at lag 12.
  - The MA(q) model: this model can be identified by observing the autocorrelation function. In fact, if there are no significant autocorrelations after lag q then MA(q) is suggested. On the other hand, if there are no significant partial autocorrelations at lag p then AR(p) may be appropriate.
  - Instances where a clear AR(p) or MA(q) does not exist. In such instances, a mixed model is suggested.

In regression analysis, the Akaike information criterion (AIC) is used to select the predictors. The same measure can also be used to select the order of ARIMA models. The AIC is calculated as

$$\text{AIC} = -2\log(L) + 2(p + q + k + 1),$$

where L is the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$. c is the intercept of the ARIMA model.

2.2.6 Behaviour of ACF and PACF for Pure AR and Pure MA Models

Table 2.1 compares the behaviour of the ACF and the PACF for pure AR and pure MA models.

2.3 Artificial Neural Network (ANN)

An artificial neural networks (ANN) model is an advanced nonlinear method that is used for forecasting. In recent years, a number of research papers have been published on the application of ANN models to time series data. These include Kaur, Kumar, and Segal (2016), Panapakidis and Dagoumas (2016), and Keles et al. (2016). In their study,

Table 2.1: Behaviour of the ACF and the PACF for pure AR and pure MA models

Process	ACF	PACF
AR(p)	Exponential decay or damped sine-wave. The exact pattern depends on the signs and sizes of $\phi_1, \ldots, \phi_p$	Spikes at lags 1 to p, then cuts off to zero
MA(q)	Spikes at lags 1 to q, then cuts off to zero	Exponential decay or damped sine-wave. The exact pattern depends on the signs and sizes of $\theta_1, \ldots, \theta_q$

Keles et al. (2016) found that an ANN model produced better results than other models in predicting day-ahead electricity spot prices. Kaur, Kumar, and Segal (2016) used an ANN model to forecast wind speed which is an essential aspect when calculating wind energy that will be produced. The model achieved the 70% accuracy mark that was set by the Central Electricity Regulatory Commission (CERC) of India.

There are two main network designs for the ANN method, namely feed-forward networks and recurrent networks. The feed-forward network design has all the arcs of the network pointing forward while the recurrent network allows for information to be fed back to the previous nodes in the network. This means that information is allowed to loop back between layers. In this section, the feed-forward network architecture is described. There are a number of variants of the ANN model which use this type of network design. The following variants, which are more suitable for time series forecasting, are described: time-lagged neural networks and seasonal artificial networks.

2.3.1 ANN Architecture

Figure 2.4 shows the architecture of a feed-forward network ANN model. This feed-forward network model is called a multi-layer perceptron (MLP) and is the ANN model most widely used for forecasting.

The model uses a network consisting of three layers, namely the input, hidden and output layers, which are connected by acyclic links. This type of network is called a feed-forward network (FNN).

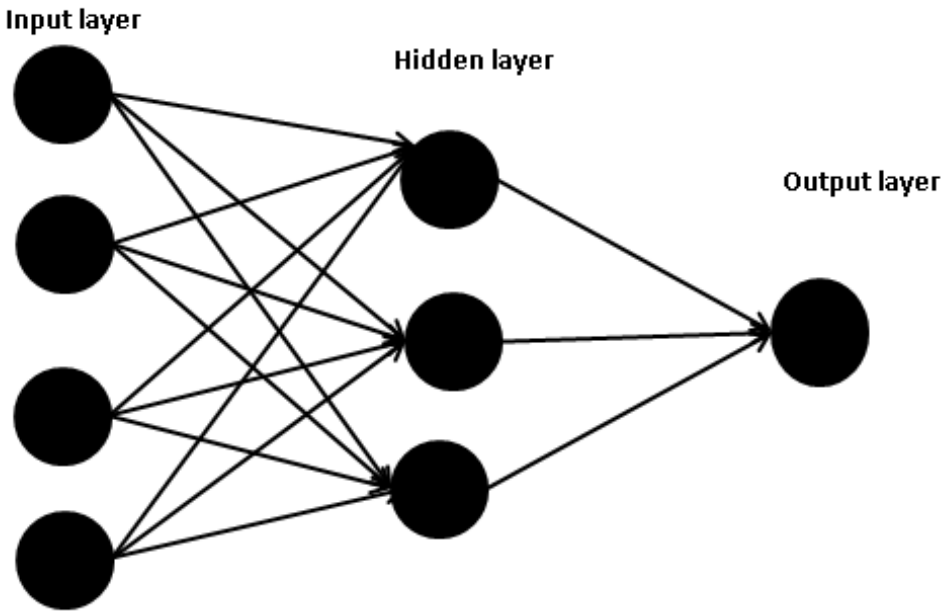


Figure 2.4: An ANN architecture with a single hidden layer

As input, each node gets a weighted sum of the nodes in the previous layer. Then a transfer function is applied to the input nodes and the relevant output is weighted to produce the final output. The mathematical expression

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g\left(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-i}\right) + \epsilon_t, \text{ for all } t \tag{2.14}$$

is used to compute the output layer of the model, where $y_{t-i}(i = 1,2,\ldots,p)$ are the p inputs and y_t is the output. The number of input and hidden nodes are represented by p and q respectively. The connection weights for the arcs are represented by α_j and β_{ij} and ϵ_t is the random shock, which is referred to as bias in the literature. The terms α_0 and β_{0j} are the intercepts terms.

2.3.2 Time-lagged Neural Networks (TLNN)

This FNN model uses the values of the time series at a particular lag. There is also an additional constant term that is connected to all the nodes in the hidden and the output

layer. This constant term is taken to be one and helps avoid introducing a bias term separately.

Figure 2.5 shows an example of an TLNN model with one hidden layer.

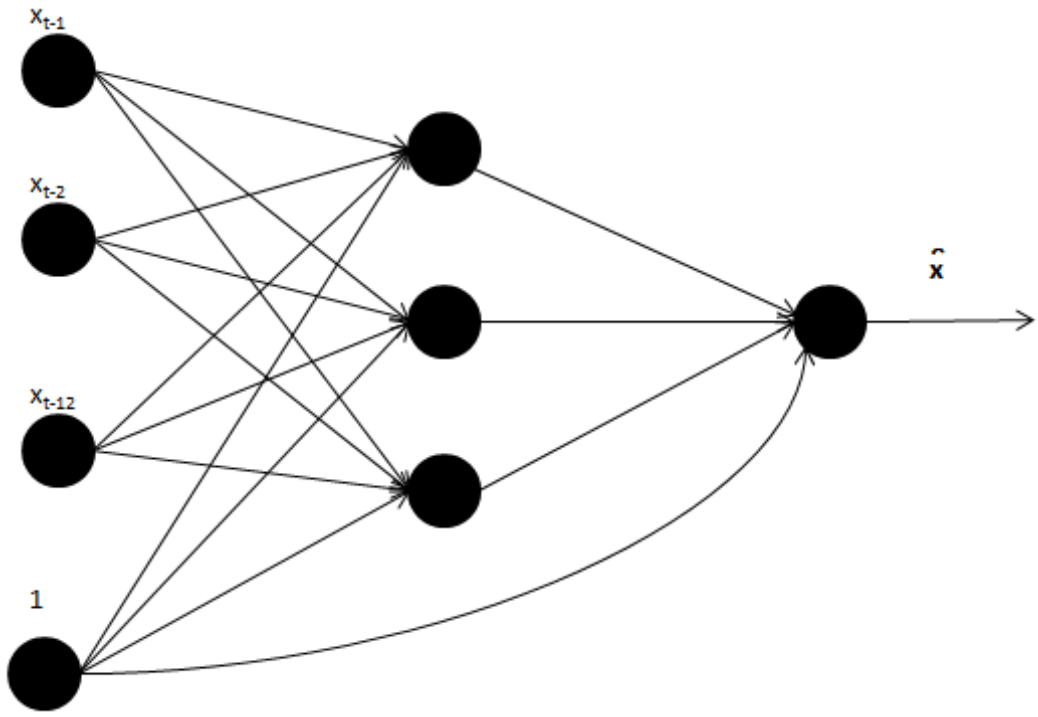


Figure 2.5: An example of a TLNN model

Each node in the hidden layer gets as input a weighted sum of the nodes in the input layer plus an additional input from the constant node. Then a transfer function is applied to the outputs from the hidden nodes. The constant node is also linked directly to the final output node. The equation

$$\hat{x}_t = \phi_0 \left\{ w_{c0} + \sum_h w_{h0} \phi_h \left(w_{ch} + \sum_i w_{ih} x_{t-j_i} \right) \right\} \tag{2.15}$$

is used to calculate the output values of the output layer for a TLNN model with one hidden layer. Here $x_{t-j_1}, x_{t-j_2}, \ldots, x_{t-j_k}$ are the input terms, w_{ch} are the weights that connect the input and the hidden layers, the w_{c0} is the connection between the constant term and the output layer, w_{ih} represents the weights for the connections between the

input and the hidden layers, and w_{h0} denotes the weights for the connections between the hidden and the output layers. ϕ_h and ϕ_0 are the activation functions for the hidden and the output layers respectively (Faraway and Chatfield, 1998).

2.3.3 Seasonal Artificial Neural Networks (SANN)

The SANN model is capable of learning the seasonal pattern in the time series without actually removing it. This model was first introduced by Hamzaçebi (2008) and is used to improve the performance of an ANN model for seasonal time series. Figure 2.6 shows a diagram of a SANN model where the input layer contains s input nodes and s is the length of the season in the time series. There is also an additional constant node as before. The number of the output nodes is also determined by length of the season s. A SANN model also uses the feed-forward architecture, but it differs from the TLNN

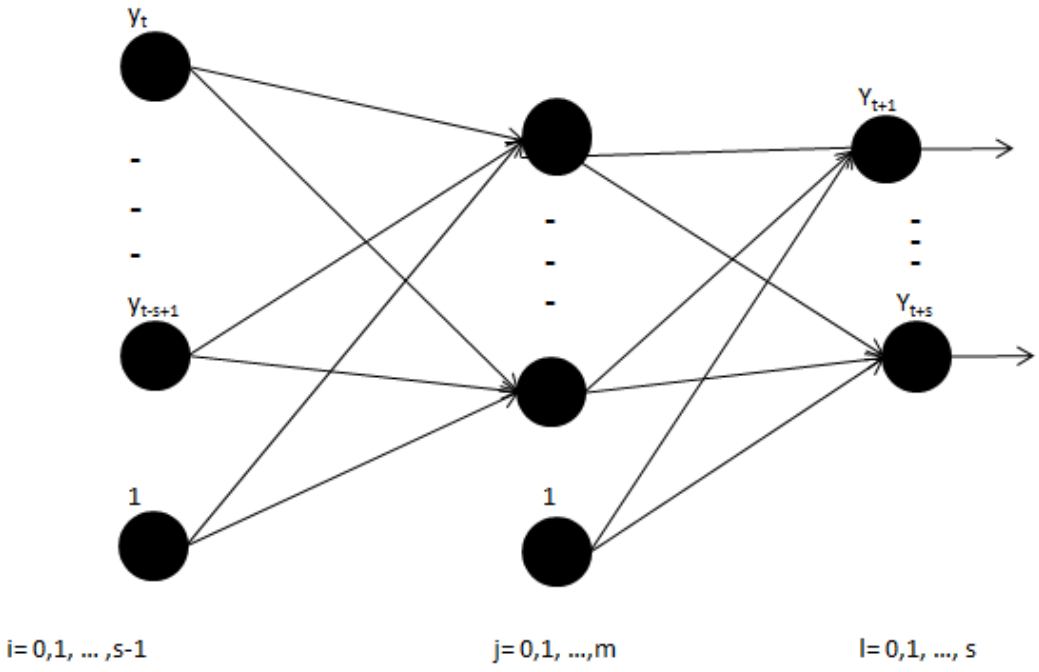


Figure 2.6: An example of a SANN model

when it comes to the inputs and outputs of the network. With SANN, the inputs are strictly the last s observations and the output is a vector of s outputs where s is the

length of the season in a seasonal time series. The output for the model is computed using the following formula:

$$Y_{t+l} = \alpha_l + \sum_{j=1}^{m} w_{jl} f\left(\theta - j + \sum_{i=0}^{s-1} v_{ij} Y_{t-i}\right), \quad \text{for all } t;\ l = 1, 2, 3, \ldots, s, \tag{2.16}$$

where Y_{t+l} is the prediction for the future s and $Y_{t-i} \quad (i = 0, 1, 2, \ldots, s-1)$ are the observations of the previous s periods. The weights from the input nodes to the hidden nodes and the hidden nodes to the output nodes are represented by v_{ij} and w_{jl} respectively. The weights of the bias connection are represented by θ_j and α_l, and f is the activation function.

2.4 Support Vector Machines

The support vector machines (SVM) method was developed by Vapnik (1995). The method was initially developed for classification problems. Owing to its success in classification, the method was also adapted for regression problems and later for time series forecasting. The SVM enjoys interest across industries, from financial markets to general business applications. According to a survey by Sapankevych and Sankar (2009), the SVM model has been applied in financial markets more than any other industry. SVM has proven to be superior to other methods when applied to financial data (Francis, Tay, and Cao, 2001; Kim, 2003; Cao and Tay, 2001). Kim (2003) used SVM and back propagation (neural networks) models to predict the direction of the daily price change in the Korean stock price index. The results showed that SVM produced better results. This suggests that the SVM could be used as an alternative to forecasting financial time series.

The SVM model uses the structural risk minimisation principle which aims at minimising the upper bound of the generalisation error. This may be compared to the traditional neural networks which use an empirical risk minimisation principle. The advantage of using SVM is that it is able to generalise to unseen data. The solution found by SVM

is always unique and optimal and this therefore guarantees that the solution found is not a local minimum. In order to understand and be able to use SVM, one needs to understand its foundations, some of which are discussed below.

2.4.1 Support Vector Regression (SVR)

In order to use the SVM for time series analysis, the concept of support vector regression (SVR) needs to be understood. The method is discussed by Vapnik (1998) and uses an ϵ-intensive loss function. The idea is to penalise errors that lie outside the ϵ-tube created. This tube is formed symmetrically around the estimated function with a minimal radius.

$$L_\epsilon(y,f(\mathbf{x},\mathbf{w})) = \begin{cases} 0 & \text{if } \mid y - f(\mathbf{x},\mathbf{w}) \mid \leq \epsilon \\ \mid y - f(\mathbf{x},\mathbf{w}) \mid -\epsilon & \text{otherwise,} \end{cases} \tag{2.17}$$

where $L_\epsilon(y,f(\mathbf{x},\mathbf{w}))$ is referred to as a ϵ-intensive loss function and, y and $f(\mathbf{x},\mathbf{w})$ are the actual and predicted values respectively.

Small values for 2.17 are desirable as they minimise the error of misclassification. This leads to the formulation of the empirical risk function in 2.18. The empirical risk function is

$$R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L_\epsilon(y_i,f(\mathbf{x},\mathbf{w})). \tag{2.18}$$

Here N is the size of the training set.

In order to minimise the risk, the following quadratic programming equations are solved:

$$\begin{aligned} \text{Minimise} \quad & J(\mathbf{w},\xi,\xi^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - \mathbf{w}^T\varphi(\mathbf{x}_i) - b \leq \epsilon + \xi_i; \text{ for all} \quad i = 1, 2, \ldots, N \\ & \mathbf{w}^T\varphi(\mathbf{x}_i) + b - y_i \leq \epsilon + \xi_i^* \\ & \xi_i \geq 0 \\ & \xi_i^* \geq 0, \end{aligned} \tag{2.19}$$

where $\varphi(\mathbf{x}_i)$ is the mapping function and b is the bias term. ξ_i and ξ_i^* are the slack variables. The slack variables represent the distance from the actual observations to the boundary of the ϵ-tube.

In order to solve quadratic programming problem 2.19, two sets of Lagrange multipliers are used namely: $\boldsymbol{\alpha} = (\alpha_1,\alpha_2,\ldots,\alpha_N)^T$ and $\boldsymbol{\alpha}^* = (\alpha_1^*,\alpha_2^*,\ldots,\alpha_N^*)^T$ where $0 \leq \alpha_i,\alpha_i^* \leq C$.

The optimal solution to 2.19 is given by

$$y(\mathbf{x}) = \sum_{i=1}^{N_s}(\alpha_i - \alpha_i^*)k(\mathbf{x},\mathbf{x}_i) + b_{opt}. \tag{2.20}$$

2.4.2 Least Square Support Vector Machine (LS-SVM)

The least square version of the SVM was formulated by Suykens and Vandewalle (1999) and uses the equality constraint instead of the inequality constraint employed by equation 2.19. It employs a sum-squared error (SSE) cost function, instead of the quadratic program used in traditional SVM. The optimisation problem is defined as:

$$\begin{aligned} &\text{Minimise} \quad \underset{w,b,e}{J(\mathbf{w},e)} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\gamma\sum_{i=1}^{N} e_i^2 \\ &\text{subject to} \quad y_i - \mathbf{w}^T\varphi(\mathbf{x}_i) + b + e_i;\ \forall_i = 1,2,\ldots,N, \end{aligned} \tag{2.21}$$

where γ is the regularisation parameter and φ is a linear mapping to a higher dimension.

The Lagrangian form can be written as:

$$L(\mathbf{w},b,e,\alpha) = J(w,e) - \sum_{i=1}^{N}\alpha_i\{w^T\varphi(\mathbf{x}_i) + b + e_i - y_i\}, \tag{2.22}$$

where $\alpha = [\alpha_i,\alpha_2,\ldots,\alpha_N]^T$ and $\alpha \leq 0$ are the Lagrange multipliers.

In order to simplify the quadratic programming problem 2.22, partial derivatives of L are derived by applying conditions of optimality. The following system of equations is

derived from the partial derivatives:

$$\begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1}^T & \Omega + \gamma^{-1}\mathbf{I} \end{bmatrix}_{(N+1)\times(N+1)} \begin{bmatrix} b \\ \alpha \end{bmatrix}_{(N+1)\times 1} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}_{(N+1)\times 1}, \tag{2.23}$$

where Ω is the kernel matrix and $\mathbf{I}$ is an $N \times N$ identity matrix. The decision function is given by

$$y(\mathbf{x}) = \sum_{i-1}^{N} \alpha_i k(\mathbf{x},\mathbf{x_i}) + b, \tag{2.24}$$

where α and b are the solutions to the matrix in 2.23.

2.4.3 Dynamic Least Squares Support Vector Machine (DLS-SVM)

The dynamic least squares support vector machine (DLS-SVM) is derived from the LS-SVM. The method was presented by Fan, Li, and Song (2006) and is highlighted here. According to Fan, Li, and Song (2006), the method works well on time series data and real-time systems. The goal of the DLS-SVM procedure is to ensure that the model adjusts to the nonlinear dynamics in the data over time. This is achieved by removing older observations from the training data and replacing them with the latest observations when they become available and the model is refined accordingly. In order to derive the DLS-SVM method the following process is followed:

From 2.23, take

$$\mathbf{Q}_N = \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1}^T & \mathbf{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix}_{(N+1)\times(N+1)}. \tag{2.25}$$

After adding the next observation to $\mathbf{Q}_N$, it becomes

$$\mathbf{Q}_{N+1} = \begin{bmatrix} \mathbf{Q}_N & \mathbf{k}_{N+1} \\ \mathbf{K}_{N+1}^T & k_{N+1}^* \end{bmatrix}_{(N+2)\times(N+2)}, \tag{2.26}$$

where $k_{N+1}^* = \gamma^{-1} + k(\mathbf{x}_{N+1},\mathbf{x}_{N+1})$ and $\mathbf{k}_{N+1} = [1,k(\mathbf{x}_{N+1},\mathbf{x}_i)]^T$.

By applying the matrix inversion lemma for $\mathbf{Q}_{N+1}^{-1}$, the following is obtained:

$$\mathbf{Q}_{N+1}^{-1} = \begin{bmatrix} \mathbf{Q}_N^{-1} + \mathbf{Q}_N^{-1}\mathbf{k}_{N+1}\mathbf{k}_{N+1}^T\mathbf{Q}_N^{-1}\rho^{-1} & -\mathbf{Q}_N^{-1}\mathbf{k}_{N+1}\rho^{-1} \\ -\mathbf{K}_{N+1}^T\mathbf{Q}_N^{-1}\rho^{-1} & \rho^{-1} \end{bmatrix}. \tag{2.27}$$

Here $\rho = k_{N+1}^* - \mathbf{K}_{N+1}^T\mathbf{Q}_N^{-1}\mathbf{k}_{N+1}$. This recursion equation removes all the matrix inversions.

When adding a new observation, the oldest observation is pruned. The first data point is removed from the training set by assuming that the new data point has just been added and that $\mathbf{Q}_{N+1}^{-1}$ is already known. The rearranged training data become $[\mathbf{x}_2,\ldots,\mathbf{x}_N,\mathbf{x}_{N+1},\mathbf{x}_1]$ and get

$$\hat{\mathbf{Q}}_{N+1} = \begin{bmatrix} \hat{\mathbf{Q}}_n & \mathbf{k}_1 \\ \mathbf{K}_1^T & k_1^* \end{bmatrix}, \tag{2.28}$$

where $k_1^* = \gamma^{-1} + k(\mathbf{x}_1,\mathbf{x}_1) = \gamma^{-1} + 1$, $\mathbf{k}_1 = [1,k(\mathbf{x}_1,\mathbf{x}_1)]^T$, for $i = 1,2,\ldots,N+1$.

$$\hat{\mathbf{Q}}_N = \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \mathbf{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix}_{(N+1)\times(N+1)} \tag{2.29}$$

with $\mathbf{\Omega}_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$, $i,j = 2,3,\ldots,N+1$. It can now be seen that the only difference between $\mathbf{Q}_{N+1}$ and $\hat{\mathbf{Q}}_{N+1}$ is the order of the rows and columns. The same applies to $\mathbf{Q}_{N+1}^{-1}$ and $\hat{\mathbf{Q}}_{N+1}^{-1}$. Therefore, by adjusting the positions of the elements of $\mathbf{Q}_{N+1}^{-1}$ the matrix $\hat{\mathbf{Q}}_{N+1}^{-1}$ can be obtained and again using the matrix inversion lemma,

$$\hat{\mathbf{Q}}_{N+1}^{-1} = \begin{bmatrix} \hat{\mathbf{Q}}_N & \mathbf{k}_1 \\ \mathbf{k}_1^T & k_1^* \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Q}_{(N+1)\times(N+1)}^* & \mathbf{P}_{(N+1)\times 1} \\ \mathbf{P}^T & q \end{bmatrix}. \tag{2.30}$$

$\hat{\mathbf{Q}}_N^{-1}$ is obtained as

$$\hat{\mathbf{Q}}_N^{-1} = \mathbf{Q}^* - \frac{\mathbf{P}\mathbf{P}^T}{q} \tag{2.31}$$

from equations 2.29 and 2.30.

The α and b from 2.23 can now be computed with $\mathbf{Q}_N^{-1}$. Then substitute the old data points with the new ones until all the new data points have been added. The values of α and b are computed from the N training data points. The following steps will be repeated for every new data point $(\mathbf{x}_{n+1}, Y_{n+1})$ for time series forecasting:

1. Compute $\mathbf{Q}_{n+1}^{-1}$.
2. Adjust the element position of $\mathbf{Q}_{n+1}^{-1}$ to get $\hat{\mathbf{Q}}_{n+1}^{-1}$ and partition $\hat{\mathbf{Q}}_{n+1}^{-1}$ into $\mathbf{Q}^*$, $\mathbf{P}$ and q.
3. Compute $\mathbf{Q}_n^{-1}$ from 2.31 to delete the first point from the training data set.
4. Compute α and b with $\mathbf{Q}_n^{-1}$ and form the LS-SVM model function

$$y(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k(\mathbf{x}, \mathbf{x}_k) + b.$$

5. Apply the new LS-SVM model function for the next time prediction.
6. Obtain a new point $(\mathbf{x}_{n+2}, Y_{n+2})$. Shift the point as $\mathbf{x}_i = \mathbf{x}_{i+1}$, $y_i = y_{i+1}$. $\mathbf{Q}_N^{-1} = \hat{\mathbf{Q}}_N^{-1}$ and repeat the steps from 1.

2.5 Regression with ARIMA Errors

In section 2.2.2 an AR(p) model was discussed. If Y_{t-i} for $i = 1, 2, \ldots p$ are replaced by $X_{i,t}$ where $X_{i,t}$ are independent variables then Y_t becomes a multiple regression model. Equation 2.5 can now be rewritten as

$$Y_t = c + \phi_1 X_{1,t} + \phi_2 X_{2,t} + \ldots + \phi_p X_{p,t} + e_t. \tag{2.32}$$

In multiple regression analysis the error term e_t is assumed to be an uncorrelated series. If this assumption is relaxed and correlated errors are allowed for, then equation 2.32

becomes a regression model with ARIMA errors. The error term is modelled as an ARIMA process and is now represented by N_t.

There is little literature on the application of this modelling approach. The only literature available is from Van den Bossche, Wets, and Brijs (2004). In their study Van den Bossche, Wets, and Brijs (2004) used a regression model with ARIMA errors to study the effect of laws, weather conditions, regulations and economic conditions on the frequency and severity of accidents. They found that the model accuracy was acceptable and that the model could be used to forecast the frequency and severity of accidents in Belgium.

2.5.1 Modelling Procedure

The method of ordinary least squares estimation cannot be used for this type of regression where the errors are correlated as this leads to incorrect estimates of the parameters. This problem is caused by the autocorrelated errors from the model, while the other problem is that autocorrelated errors may lead to spurious regression. Instead, a generalised least squares estimation or method of maximum likelihood estimation is recommended. By means of minimising,

$$G = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j N_i N_j, \tag{2.33}$$

generalised least squares estimates can be obtained. Here w_i and w_j are weights based on the patterns of the autocorrelations, and N_i and N_j are the correlated error terms. The method of generalised least squares estimation works only if the errors follow an ARIMA process.

The following procedure as described by Hyndman and Athanasopoulos (2018), can be used to fit a regression model with ARIMA errors:

1. Fit the regression model with a proxy AR(1) or AR(2) model for errors.

2. If the errors from the regression appear to be non-stationary, and differencing appears appropriate, then difference the forecast variable and all explanatory variables. Then fit the model using the same proxy model for errors, this time using differenced variables.

3. If the errors now appear stationary, identify an appropriate ARIMA model for the error series, N_t.

4. Refit the entire model using the new ARIMA model for the errors.

5. Check that the e_t residual series looks like white noise.

2.6 Bootstrap Aggregating

Bootstrap aggregating also known as bagging was first suggested by Breiman (1996). According to Härdle, Horowitz, and Kreiss (2003), bootstrapping is "a method for estimating the distribution of an estimator or test statistic by resampling one's data or a model estimated from the data". In time series analysis, bagging is achieved by randomly generating new time series that are similar to the original time series with the aid of a bootstrap method. With this approach, the inputs to the model are bootstrapped and copies of possible predictors are estimated from the bootstrapped inputs. An aggregate of the output of these predictors is then used as the final output of the model. This method is popular in machine learning and is used to improve the accuracy of the predictors.

The literature on the application of bagging shows some successes where the method was used. For example, Khwaja et al. (2015) compared the performance of a bagged neural network (BNN) to other models including an ARIMA and ANN model. The results proved that the BNN model was better at predicting the short-term load in the electricity grid. In their study titled "Bagging Exponential Smoothing Methods using STL Decomposition and Box-Cox Transformation", Bergmeir, Hyndman, and Benítez (2016) proposed a method for bagging exponential smoothing methods. The results suggest that the method was better than the original exponential smoothing models

when applied to the Makridakis Competitions (M3) dataset. The M3 datasets contain time series data from different industries and the time horizon used in the competition (Makridakis and Hibon, 2000). The model also achieved better results than any of the competition participants for the monthly datasets.

In this section, moving block bootstrap (MBB), dependent wild bootstrap (DWB) and tapered block bootstrap (TBB) bagging procedures are discussed.

2.6.1 Moving Block Bootstrap (MBB)

The moving block bootstrap is a bootstrap method used for resampling correlated time series observations. With the moving block bootstrap, the time series data are separated into blocks which may overlap. These blocks become observations and a number of observations are drawn at random with replacement and aligned in the order they were picked to form a bootstrap series. The method is free of assumptions and does not require intermediate computations of other quantities (Mignani and Rosa, 1995).

The MBB is described by Lahiri and Lahiri (2003) as follows: Let $\beta_i = (X_i, \ldots, X_{i+\ell-1})$ represent a series of length ℓ starting with X_i where $1 \leq i \leq N$ and $N = n - \ell + 1$. The MBB samples can be obtained by randomly selecting a suitable number of blocks from the collection $\{\beta_1, \ldots, \beta_N\}$.
Let $\{\beta_1^*, \ldots, \beta_N^*\}$ represent a simple random sample drawn with replacements from $\{\beta_1, \ldots, \beta_N\}$ and each of the blocks has ℓ elements. If the elements of β_i are represented by $(X^*_{(i-1)\ell+1}, \ldots, X^*_{i\ell})$, then $X_1^*, \ldots, X_m^*$ is the MBB sample of size $m \equiv k\ell$. The equation

$$\theta^*_{m,n} = T(F^*_{m,n})$$

is defined as the MBB version $\theta^*_{m,n}$ of $\hat{\theta}_n$. Here $F^*_{m,n}$ is the empirical distribution of $(X_1^*, \ldots, X_m^*)$.

2.6.2 Dependent Wild Bootstrap (DWB)

The DWB method was suggested by Shao (2010). This bootstrap method is suitable for stationary time series and can also be used when the time series is irregularly spaced. The method also works well on data which are weakly dependent. Unlike the MBB method, the DWB does not require the data to be partitioned into blocks.

Let $X_N = \{X_t\}_{t=1}^n$ be a set of observations with $\mu = E(X_t)$ and $\gamma_k = cov(X_0,X_k)$, then DWB pseudo-observations can be defined as

$$X_i^* = \bar{X}_n + (X_i - \bar{X}_n)W_i, \text{ for } \quad i = 1,\ldots,n; \tag{2.34}$$

where $\bar{X}_n = \frac{1}{n}\sum_{t=1}^n X_t$ and $\{W_i\}_{i=1}^n$ are the sample mean and n random variables respectively. The $\{W_i\}_{i=1}^n$ satisfy the following assumption:

- The random variables, $\{W_i\}_{i=1}^n$, are independent of the data X_n. For $t = 1,\ldots,n$ the $E(W_t) = 0$ and $\text{var}(W_t) = 1$. Assume that W_t is a stationary process with $\text{cov}(W_t,W_t') = a\{(t-t')/I\}$, where $a(.)$ is a kernel function and $I = I_n$ is a bandwidth parameter. Furthermore, assume that

$$K_a(x) = \int_{-\infty}^{\infty} a(u)e^{-iux}d\text{u} \geq 0 \text{ for } x \in \Re. \tag{2.35}$$

2.6.3 Tapered Block Bootstrap (TBB)

Politis and Paparoditis (2001) introduced the method of tapered block bootstrap (TBB) where a sequence of data-tapered windows is produced. The results produced by this method were found to be superior to those produced using the other block bootstrap methods for time series. First the data X_t, where $t = 1,2,\ldots,N$ is centred by letting $Y_t = X_t - \bar{X}_N$. Then a sequence of data-tapering windows $w_n(.)$ is introduced. The values of the weights $W_n(t)$ lie between 0 and 1. More specifically $W_n(t) = 0$ for $t \notin \{1,2,\ldots,N\}$. It follows that $\|w_n\|_1 \leq n$ and $\|w_n\|_2 \leq n^{\frac{1}{2}}$. Here $\|w_n\|_1 = \sum_{t=1}^n |w_n(t)|$ and $\|w_n\|_2 = \{\sum_{t=1}^n w_n^2(t)\}^{\frac{1}{2}}$. The sequence of data-tapering windows $w_n(.)$ can be constructed by

dilating a single function $w : \Re \to [0,1]$, so that

$$w_n(t) = w\left(\frac{t-0.5}{n}\right). \tag{2.36}$$

The function $w(.)$ is assumed to satisfy the following assumptions:

- $w(t) \in [0,1]$ for all $t \in \Re$, $w(t) = 0$ if $t \notin [0,1]$, and $w(t) > 0$ for t in the neighbourhood of $\frac{1}{2}$.
- The function $w(t)$ is symmetric about $t = 0.5$ and non-decreasing for $t \in [0,\frac{1}{2}]$. The TBB algorithm can now be defined as follows:
  - First choose a positive integer b less than N, and let $i_0,i_1,\ldots,i_{k-1}$ be drawn independently and identically distributed uniform on the set $\{1,2,\ldots,Q\}$, where $Q = N - b + 1$. Here $k = [N/b]$ is taken, where $[.]$ is the integer part.
  - For $m = 0,1,\ldots,k-1$, let

$$Y^*_{mb+j} := w_b(j)\frac{b^{\frac{1}{2}}}{\|w_b\|_2} Y_{i_m+j-1} \ (j = 1,2,\ldots,b). \tag{2.37}$$

  - Finally, construct the bootstrap sample mean $\bar{Y}^*_I = \frac{1}{I}\sum_{i-1}^{I} Y^*_i$.
- The self-convolution $w * w(t)$ is twice continuously differentiable at the point $t = 0$, where $w * w(t) = \int_{-1}^{1} w(x)w(x+|t|)\mathrm{d}x$.

2.7 Application to Aviation and Comparison of Methods

Time series forecasting is widely used in aviation to forecast air passenger demand. The application of time series methods to forecast this demand in the literature includes studies by Yukun, Tao, and Zhongyi (2012), Adeniran, Kanyio, and Owoeye (2018), Dantas, Cyrino Oliveira, and Varela Repolho (2017), and Weatherford, Gentry, and Wilamowski (2003). Yukun, Tao, and Zhongyi (2012) proposed an ensemble empirical mode decomposition (EEMD)-Slope-SVMs; this modelling approach is based on the SVM modelling framework. They used air passenger data from selected airlines in the

United Kingdom and the United States for comparing the use of the SVM, Holt-Winters and ARIMA methods. Adeniran, Kanyio, and Owoeye (2018) used single moving average and simple exponential smoothing methods to forecast demand for air passengers in Nigeria. Dantas, Cyrino Oliveira, and Varela Repolho (2017) combined bootstrap aggregating (bagging) and Holt Winters methods to forecast demand for air passengers using data from 14 different countries. An application of neural network forecasting in an airline can be found in Weatherford, Gentry, and Wilamowski (2003). They compared the method with traditional forecasting methods and the results showed promising performance.

A comparison of forecasting methods can be found in Ahmed et al. (2010) and Makridakis, Spiliotis, and Assimakopoulos (2018). Ahmed et al. (2010) compared the performance of machine learning methods and found that multi-layer perceptron and the Gaussian process regression performed better than the other methods. The machine learning methods compared included Bayesian neural networks, radial basis functions, generalised regression neural networks, K-nearest neighbour regression, CART regression trees and support vector regression. In the comparison, the traditional statistical methods for forecasting were not considered which makes it difficult to generalise the performance of the methods. Makridakis, Spiliotis, and Assimakopoulos (2018) compared the performance of the traditional statistical forecasting methods with the machine learning methods. The results showed that the traditional statistical methods outperformed machine learning methods and the authors concluded that there is still a lot of work required to improve machine learning methods for forecasting.

2.8 Advantages and Disadvantages of the Different Methods

In this section, the advantages and disadvantages of the different forecasting methods are discussed. Accordingly, the literature is examined in order to understand these factors.

2.8.1 Advantages

ARIMA models are not particularly sensitive to the underlying assumptions of the nature of the data fluctuations, but they are highly accurate in forecasting short time horizons. Under special conditions, these models are equivalent to other models like exponential smoothing. Bagging is a method which is very useful in improving the accuracy of the underlying method and the regression model with ARIMA errors will inherit the strengths of the ARIMA method.

ANN models are capable of detecting nonlinear and other complex relationships in the data. This method is able to derive those relationships without any prior assumptions regarding the relationships. This makes ANN models some of the most used models in prediction problems. The ANN models are also able to derive meaning from imprecise data when the relationship among the variables is difficult to explain using conventional methods (Zhang and Qi, 2005).

Most of the advantages that are highlighted for ANN models also apply to SVM models. In addition, SVM models are not prone to the problem of overfitting suffered by ANN models. Compared to ANN models, SVM models in general provide better results.

2.8.2 Disadvantages

The ARIMA model needs many observations in order to accurately capture the components of the time series. There is no automatic updating of the model, instead the model will have to be refitted with new observations. The procedure for selecting parameters during the identification stage is subjective and the method is best only for forecasting short time horizons. The method also requires that the data should be stationary in variance, otherwise transformation is required.

ANN models are sometimes criticised for their "black box" nature. This is because even though better results can be obtained with this method, the results cannot be explained. ANN models also tend to be prone to overfitting and they are also data intensive compared to other forecasting methods. Training an ANN model requires large quantities of

data which in turn demands a large amount of computer memory (Zhang and Qi, 2005). For the SVM, when training data is large the computation time increases. Even though the choice of correct hyper-parameters is critical in the SVM, there is still no structured way of choosing the correct hyper-parameters in advance.

Bagging cannot be used as a forecasting method alone, it can only be used in conjunction with other forecasting methods in order to improve the accuracy of a forecasting method. The method may also be computationally expensive.

The advantages and disadvantages of the different methods are summarised in Table 2.2.

2.9 Implementing the Forecasting Methods in R

R is open source software used for statistical computing. One of the reasons why the software is favoured is because of the ease with which high quality plots can be produced. R also allows users to develop their own packages or extensions and share them with the community. This makes it easier for new statistical methods to find their way to R users faster than other statistical software (R Core Team, 2017).

For the ARIMA models and all the other advanced methods considered here, except SVM, the *forecast* package in R developed by Hyndman et al. (2018) offers functions to fit the different models. The $nnetar()$ function is used to fit an ANN model. The function fits an FFN with one hidden layer as explained in Section 2.3. The function $nnetar()$ requires that the time series be stationary in the variance. This enables the function to achieve better results.

The function $Arima()$ is suitable for fitting ARIMA models in R. The parameters *order* and *seasonal* are some of the most important parameters in the function, as *order* specifies the order of the ARIMA model while *seasonal* specifies the order of the seasonality. Using the function $auto.arima()$, allows for the automatic selection of a suitable ARIMA model for a given time series. In choosing a suitable ARIMA model, the $auto.arima()$ function follows the methodology explained in Section 2.2. The parameter $lambda$ is

Table 2.2: Advantages and disadvantages of the methods

Method	Advantages	Disadvantages
ARIMA	Not very sensitive to the underlying assumptions of the nature of the data fluctuations Highly accurate in forecasting short time horizons Under special conditions, these models are equivalent to other models like exponential smoothing	Data intensive No automatic updating of the model The procedure for selecting parameters during the identification stage is subjective Data should be stationary in variance
ANN	Capable of detecting nonlinear and other complex relationships in the data Able to derive meaning from imprecise data	"Black box" nature Require large quantities of data Demand large amounts of computer memory Prone to overfitting
SVM	Not prone to the problem of overfitting	No structured way of choosing correct hyper-parameters Tuning the hyper-parameters is time-consuming
Regression with ARIMA errors	Not very sensitive to the underlying assumptions of the nature of the data fluctuations Highly accurate in forecasting short time horizons Under special conditions, these models are equivalent to other models like exponential smoothing	Data intensive No automatic updating of the model The procedure for selecting parameters during the identification stage is subjective Data should be stationary in variance Forecasts are also required for the predictor variable
Bagging	Improves the accuracy of the underlying method The method may also be computationally expensive	Used in conjunction with other forecasting methods

very useful for a time series that is not stationary in the variance, as this parameter allows for both the automatic selection and the manual specification of the lambda value. The estimated lambda is used to transform the time series using BoxCox transformation

(Hyndman and Athanasopoulos, 2018). The formula

$$w_t = \begin{cases} \ln y_t & \text{if } \lambda = 0; \\ \frac{(y_t^{\lambda} - 1)}{\lambda} & \text{otherwise,} \end{cases} \tag{2.38}$$

where w_t is the BoxCox transformed series, is used to transform a non-stationary time series to one that is stationary in the variance. In R, the function $BoxCox()$ from the *forecast* package can be used to transform the time series.

The transformed time series can also be reversed using the formula

$$y_t = \begin{cases} \exp(w_t) & \text{if } \lambda = 0; \\ (\lambda w_t + 1)^{\frac{1}{\lambda}} & \text{otherwise.} \end{cases} \tag{2.39}$$

The function $boxcox.inv()$ in the *forecast* package can be used for this purpose.

Hyndman and Athanasopoulos (2018) highlight that the back-transformed forecasts obtained from equation 2.39 are usually not the mean of the forecast distribution but rather the median of the forecast distribution. The equation

$$y_t = \begin{cases} \exp(w_t)\left[1 + \frac{\sigma_h^2}{2}\right] & \text{if } \lambda = 0; \\ (\lambda w_t + 1)^{\frac{1}{\lambda}}\left[1 + \frac{\sigma_h^2(1-\lambda)}{2(\lambda w_t + 1)^2}\right] & \text{otherwise,} \end{cases} \tag{2.40}$$

can be used in cases where mean point forecasts are required. Here σ_h^2 is the h-step forecast variance. The difference between the results obtained from equations 2.39 and 2.40 is known as bias.

For SVM, the function $svm()$ in the *e1071* package is used to estimate the parameters of the forecasting model. The R package was written by Meyer et al. (2018). The *e1071* package implements the SVR model discussed in Section 2.4. The success of the method depends on the choice of the parameters cost (C) and gamma (γ). The *e1071* package allows for the tuning of these parameters using a grid of a number of combinations

by using the *tune.svm*() function. Hsu, Chang, and Lin (2016) suggest a combination of values in the range $[2^{-5},2^{15}]$ and $[2^{-15},2^{3}]$ for C and γ respectively. The function optimises the values by maximising the information criteria.

The bootstrap aggregating method discussed by Makridakis and Hibon (2000) is implemented in the *forecast* package. The modelling procedure uses the moving block bootstrap discussed in Section 2.5. The function *bld.mbb.bootstrap*() in R can be used to bootstrap a time series. The most important arguments in this function are the time series x and the number of bootstrapped versions to be generated (num). To produce forecasts from this, an average of the bootstraps at time t is calculated resulting in time series of means. Forecasts can now be produced using the *ets*() function as usual. The *baggedETS*() function can also be used to fit this kind of model.

The regression model with ARIMA errors uses the same function as the ordinary ARIMA model. The only difference lies in the arguments where *xreg* is specified for the regression model with ARIMA errors. In order to produce forecasts for the model, the forecasts for the *xreg* variable need to be inferred or forecast first for the forecast horizon.

Chapter 3

Accuracy Measures

3.1 Introduction

In this chapter, the different accuracy measures applied to forecasting models are investigated.

3.2 Standard Statistical Measures

All models should be evaluated in order to determine how well they are able to predict future values for a given dataset. In this section, a number of forecasting accuracy measures are introduced and discussed.

The error of a forecast e_t is defined as the difference between the actual value of the series Y_t and the forecast value F_t. This is represented as

$$e_t = Y_t - F_t. \tag{3.1}$$

F_t is called the one-step-ahead forecast and is calculated using the past observations $Y_1, \ldots, Y_{t-1}$. If there are n observations and forecasts, then there will be n error terms and some statistical measures could be calculated.

3.2.1 Scale-dependent Accuracy Measures

Using equation 3.1 one can compute the errors for each period. An average of these errors can be calculated using

$$ME = \frac{1}{n}\sum_{t=1}^{n} e_t. \tag{3.2}$$

Equation 3.2 is called the mean error (ME). Since the errors will wander between positive and negative, ME will always be closer to zero. This means that ME is not a good measure of accuracy.

In order to try to overcome the problem of negative and positive errors suffered by ME, mean absolute error (MAE) and mean squared error (MSE) are defined as

$$MAE = \frac{1}{n}\sum_{t=1}^{n} \mid e_t \mid \tag{3.3}$$

and

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{3.4}$$

respectively. The MAE is easy to interpret and explain to a non-technical person. By squaring the errors, MSE ensures that all the errors become positive and the mean of these squared errors is then calculated. The MSE is much easier to handle mathematically compared to MAE. Taking the square root of the MSE results in another popular measure of accuracy called the root mean squared error (RMSE). The RSME is represented mathematically by the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}. \tag{3.5}$$

3.2.2 Percentage Accuracy Measures

The main weakness of the accuracy measures considered so far is that their size depends on the scale of the observations. The next set of accuracy measures are relative measures. These measures make it easier to compare forecast accuracy across time frames and

different time series. The mean percentage error (MPE) is calculated as

$$MPE = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{Y_t - F_t}{Y_t}\right) \times 100. \tag{3.6}$$

Like the ME, the MPE suffers from the effects of negative and positive errors. Another relative measure is called the mean absolute percentage error (MAPE) and is defined as

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\left(\frac{Y_t - F_t}{Y_t}\right)\right| \times 100. \tag{3.7}$$

3.2.3 Scaled Accuracy Measures

As an alternative to using percentage errors, Hyndman and Koehler (2006) proposed scaled errors. For a non-seasonal time series, the scaled error is defined as

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}. \tag{3.8}$$

For seasonal time series, a scaled error can be calculated as

$$q_t = \frac{e_t}{\frac{1}{n-s}\sum_{i=s+1}^{n}|Y_i - Y_{i-s}|}. \tag{3.9}$$

The scaled measures can now be calculated from the scale-dependent measures. For example, the mean absolute scaled error (MASE) is simply

$$MASE = \frac{\sum_{t=1}^{N}|q_t|}{N}. \tag{3.10}$$

3.2.4 Relative Accuracy Measures

It is sometimes important to compare a forecasting model with a baseline model. A baseline model can be an existing model that was developed and used before, or a "naïve model". For non-seasonal time series, a naïve model is defined as $F_t = Y_{t-1}$. This means that the forecast for the next period is the same as the most recent available

observation. This is called a naïve forecast 1 or NF1. For seasonal time series, the forecast for the next period is the same as the last observation in the corresponding season. This can be represented mathematically as $F_t = Y_{t-s}$. This simple model is called naïve forecast 2 or NF2 (Makridakis, Wheelwright, and Hyndman, 1998).

Let MAE_b be an MAE for the naïve model, then relative MAE (relMAE) is defined as MAE/MAE_b. The other measures can also be derived in the same way.

Another relative measure called Theil's U statistic, can also be used. Mathematically, Theil's U statistic is calculated as

$$U = \sqrt{\frac{\sum_{t=1}^{n-1}\left(\frac{F_{t+1}-Y_t-Y_{t+1}+Y_t}{Y_t}\right)^2}{\sum_{t=1}^{n-1}\left(\frac{Y_{t+1}-Y_t}{Y_t}\right)^2}} = \sqrt{\frac{\sum_{t=1}^{n-1}\left(\frac{F_{t+1}-Y_{t+1}}{Y_t}\right)^2}{\sum_{t=1}^{n-1}\left(\frac{Y_{t+1}-Y_t}{Y_t}\right)^2}}. \tag{3.11}$$

The following interpretations can be made of Theil's U statistic (Makridakis, Wheelwright, and Hyndman, 1998):

- When $U = 1$, then the forecasting method being evaluated is as good as the naïve method.
- When $U < 1$, then the forecasting method being evaluated performs better than the naïve method. In fact the smaller the value of U, the better the forecasting method.
- When $U > 1$, then the forecasting method under investigation performs worse than the naïve method. Therefore there is no point in using the proposed model.

The literature shows that there is a number of other forecast accuracy measures and Table 3.1 summarises some of these measures.

Table 3.1: Other accuracy measures

Accuracy Measure	Symbol	Formula
Median absolute percentage error	MdAPE	$\underset{t=1,n}{\text{median}}\left(\frac{\lvert Y_t-F_t\rvert}{Y_t}\times 100\right)$
Root mean square percentage error	RMSPE	$\sqrt{\underset{t=1,n}{\text{mean}}\left(\frac{\lvert Y_t-F_t\rvert}{Y_t}\times 100\right)^2}$
Root median square percentage error	RMdSPE	$\sqrt{\underset{t=1,n}{\text{median}}\left(\frac{\lvert Y_t-F_t\rvert}{Y_t}\times 100\right)^2}$
Symmetric mean absolute percentage error	sMAPE	$\underset{t=1,n}{\text{mean}}\left(200\bullet\frac{\lvert Y_t-F_t\rvert}{Y_t+F_t}\right)$
Symmetric median absolute percentage error	sMdAPE	$\underset{t=1,n}{\text{median}}\left(200\bullet\frac{\lvert Y_t-F_t\rvert}{Y_t+F_t}\right)$
Root mean square scaled error	RMSSE	$\sqrt{\underset{t=1,n}{\text{mean}}\left(\left(\frac{\lvert e_t\rvert}{\frac{1}{n-1}\sum_{i=2}^{n}\lvert Y_i-Y_{i-1}\rvert}\right)^2\right)}$

3.3 Out-of-sample Accuracy Measures

The acceptable results obtained during model fitting do not give a guarantee that the model will generalise, as the model might still not perform well when used with unseen future observations. For that reason, it is customary to withhold or hide the last 20% of the time series during model fitting. This set of withheld observations is called a test set whereas the rest is called the training set. The training set is used to fit an appropriate forecasting model while the test set is used to evaluate the accuracy of the model.

One of the main reasons why models perform well on the training set and bad on the test set is overfitting. Overfitting occurs when a model used to fit a training set fails to acknowledge that some variations in the data pattern are due to randomness. To demonstrate this concept consider Figure 3.1 which was produced using a function $y = ax + b$, where b is a normally distributed random number. In this demonstration, the last three observations were used as a test set. Two functions were fitted to the first nine observations as shown. The polynomial function seems to fit the data well compared to the linear model.

Figure 3.2 shows the projections of the two models with the test data. The polynomial

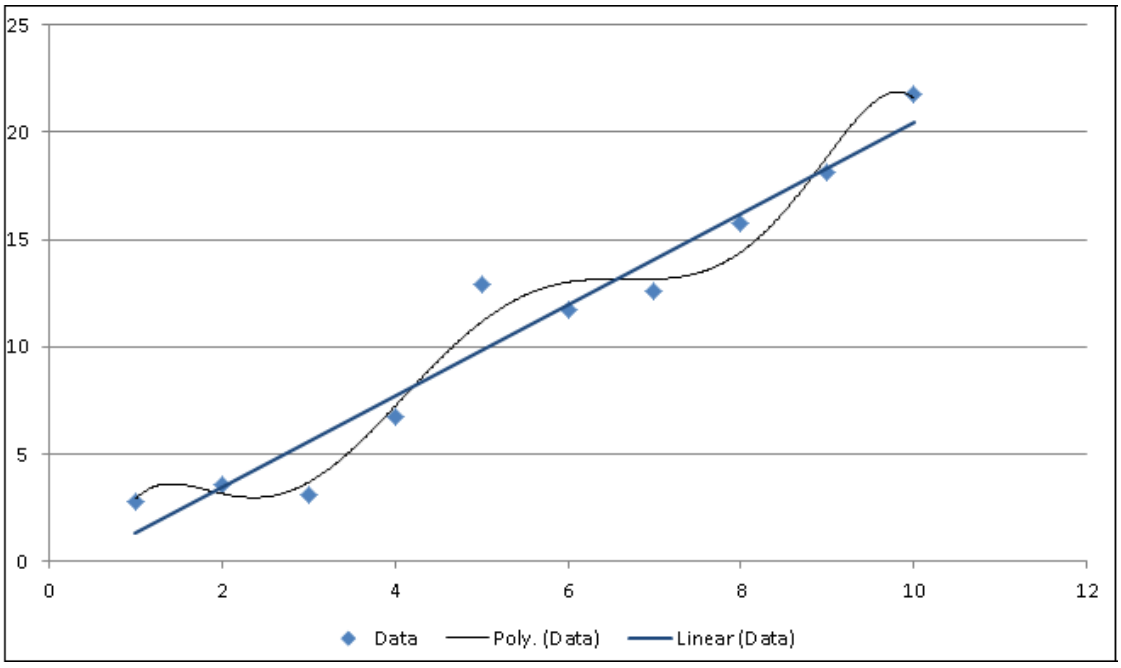


Figure 3.1: Linear versus polynomial model

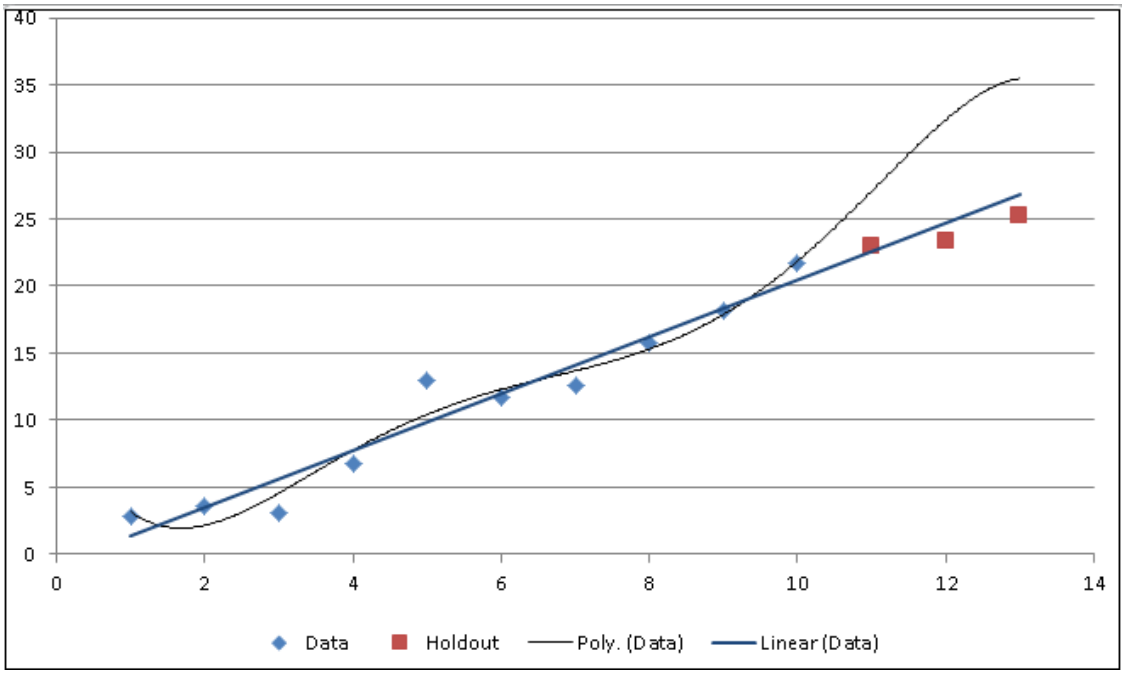


Figure 3.2: Linear versus polynomial model results

model fails to generalise to the unseen data whereas the linear model seems to perform better on the test set compared to the polynomial model. If the entire dataset were used to fit the models, an incorrect model would have been chosen. The main advantage

of having holdout data is time compression, as the forecasting model performance is determined instantly without the need to wait for new data in the future. Splitting the time series into a training and test set also enables earlier detection of overfitting. The model can be adjusted and re-tested before deployment, saving cost and precious time in the process.

3.4 Accuracy Measures in R

The *forecast* package created by Hyndman et al. (2018) in R can be used to calculate most of the accuracy measures presented. The performance of the method is shown for both the training and the test sets and the best model is determined by evaluating its performance on the test set.

Chapter 4

Application of Advanced Forecasting Methods

4.1 Introduction

This chapter focuses on the application of the different forecasting methods discussed in Chapter 2. In this chapter, the data set analysed is introduced in Section 4.2, the data are prepared for the fitting of the forecasting methods in Section 4.3, the different forecasting methods are fitted to the data and the results are analysed in Sections 4.4 to 4.8, and the findings are made in Section 4.9. In Section 4.10, the forecasting models are bagged and Section 4.11 presents the results as well as a summary.

4.2 The Data Set

The data set contains the total number of air passengers passing through major South African airports per month. This data set contains 94 observations and pertains to the period between April 2012 and January 2020, a period of seven years and 10 months. The data set was obtained from the Airports Company South Africa website (Airports Company South Africa, 2020); such data are published monthly by the company. The passenger data set is made up of four passenger components:

1. International: These are the international arrivals and departures passing through South African airports.
2. Regional: Refers to passengers arriving or departing to other countries within the Southern African Development Community (SADC) member states that go through South African airports.
3. Domestic: These are passengers within the country who pass through South African airports.
4. Unscheduled: These are air travellers whose destination or origin is unknown who travel through the airports.

Figure 4.1 shows the proportions of air passengers by the components. The majority of passengers in the data are domestic at 65%, followed by international passengers at 31%. The regional passengers make up a mere 3%.

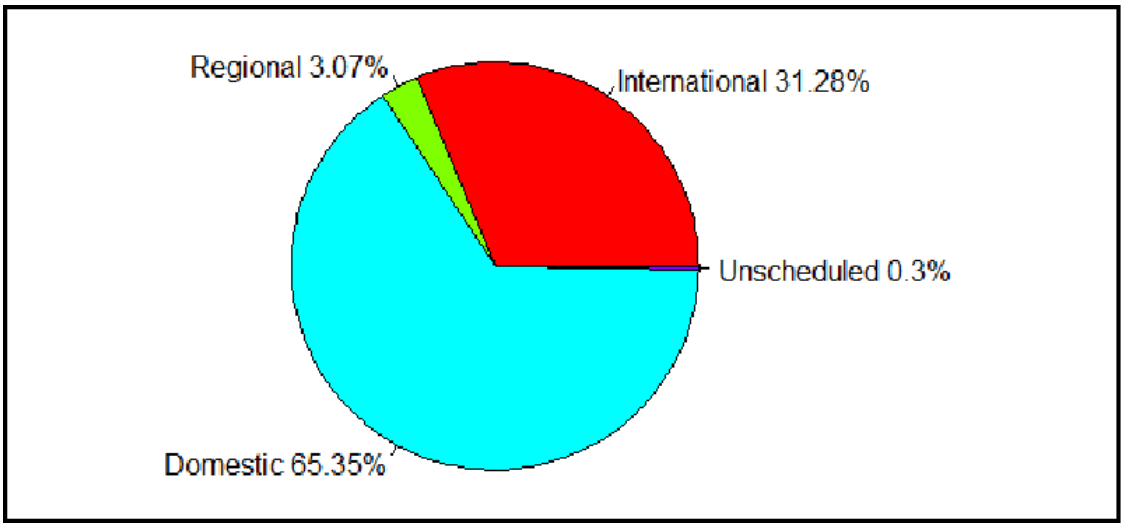


Figure 4.1: Proportion of passengers by type

4.3 Exploring the Time Series

Figure 4.2 shows a time plot of the monthly number of air passengers. The time plot shows an upward trend and signs of seasonality. It also shows that passenger numbers are at their highest for the year in December and March, and lowest in June. The peaks

correspond to the summer holiday season in the southern hemisphere while the lows correspond to the winter season. The visibility of trend and seasonality means that the series is not stationary in the mean. The slight variation in the level of seasonality over time suggests that the series may also be non-stationary in the variance.

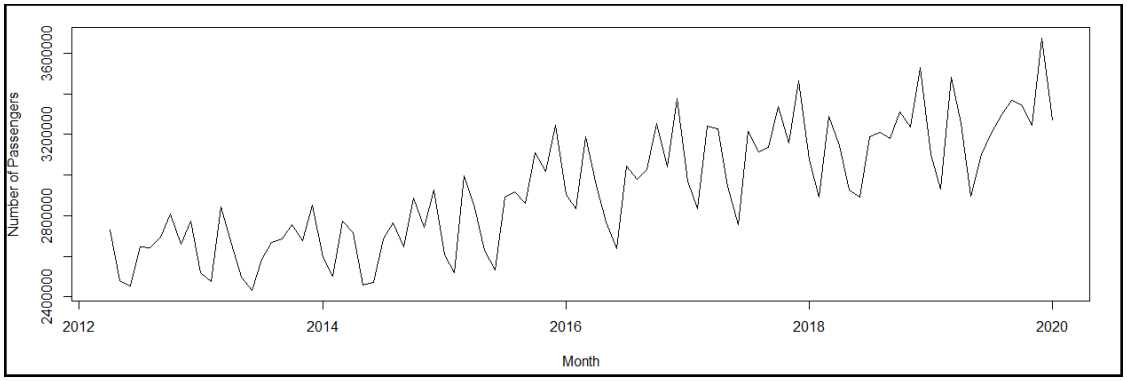


Figure 4.2: Passengers passing through South African airports from April 2012 to January 2020

The time series was broken down into its basic components and the results are shown in Figure 4.3. The plot confirms earlier observations about the behaviour of the time series. The combination of the functions *decompose()* and *plot()* in R were used to produce the results. The *decompose()* function automatically selects an optimised procedure to split the time series into trend, seasonality and irregular component using moving averages. The trend in the air passenger time series seems to evolve over time. Initially, there appears to be no trend between April 2012 and March 2014, but this is followed by a steep upward trend between April 2014 and 2018, which in turn is followed by a slightly flatter trend thereafter.

The *decompose()* function also determines the type of seasonality observed in the series. Seasonality can take on two possible forms, additive seasonality and multiplicative seasonality. The former is chosen when the time series does not show any signs of a change in variance over time, while the latter is chosen when there are signs of a change in the variance of the series over time. This can be used as an indicator for the presence of non-stationarity in the variance of a time series. For the air passenger data, additive seasonality was chosen. This indicates that the time series is stationary in the variance

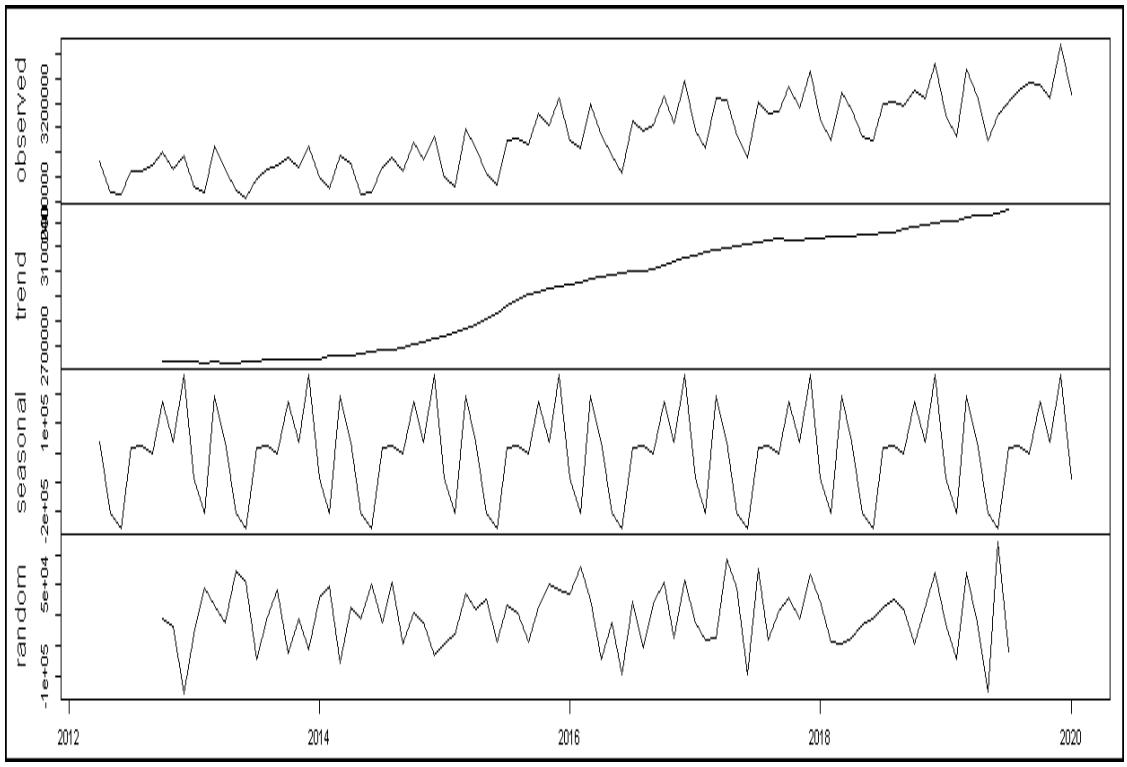


Figure 4.3: Components of the passengers time series

and there will be no need to transform the series to remove the non-stationarity in the variance.

4.4 Splitting the Time Series into Training and Test Sets

For the purpose of fitting the different forecasting methods, the last 12 months of the data were reserved as a test set. This means that February 2019 to January 2020 observations were not used during model fitting and only used later on to see how models performed on unseen data. Due to the limited number of observations, reserving the last twelve months ensured that the selected model would be tested on a full season while also maximising the available data for training the model. The data used for fitting the models will be referred to as the training set and the remaining observations will be referred to as the test set.

4.5 Fitting the ARIMA Model

A suitable ARIMA model was investigated for the time series. The function *auto.arima()* from the *forecast* package introduced in Section 2.8 was used to fit a suitable ARIMA model for the time series. The *auto.arima()* function implements the ARIMA fitting methodology explained in Section 2.2 (Hyndman et al., 2018).

For the air passenger data, an ARIMA$(0,1,1)(0,1,1)_{12}$ model was chosen with parameters indicated in Table 4.1. This meant that for this model there was no AR part. The model contains both the MA and the seasonal MA part. There is a normal differencing of order one, together with a 12-month seasonal differencing of order one.

Table 4.1: Coefficients of ARIMA$(0,1,1)(0,1,1)_{12}$ model

Parameter	Coefficient	Standard error
MA(1)	−0,6038	0,0879
Seasonal_MA(1)	−0,4723	0,1305

After substituting the variables in equation 2.9, it becomes

$$(1-\mathbf{B})(1-\mathbf{B}^{12})y_t = (1+0{,}6\mathbf{B})(1+0{,}47\mathbf{B}^{12})\eta_t. \tag{4.1}$$

Table 4.2 shows the performance measures of the fitted ARIMA$(0,1,1)(0,1,1)_{12}$ model on both the training and test data.

Table 4.2: Performance measures of the fitted ARIMA$(0,1,1)(0,1,1)_{12}$ model

Data	**Data type**	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
Air Passengers	Training set	5957,163	59604,47	46273,640	0,184	1,585	0,450	0,273
	Test set	21071,795	91643,330	81043,460	0,566	2,503	0,787	0,338

The value of MAPE is 1,585 on the training set, indicating that the fitted model can be expected to produce forecasts that are on average close to 1,6% of the actual observations. The value of MAE is 46 273,64 . Both the values of Theil's U statistic (0,273) and MASE (0,45) are below one, giving an indication that the model is better than a naïve model. The value of RMSE is also vital in establishing the usefulness of a model, and

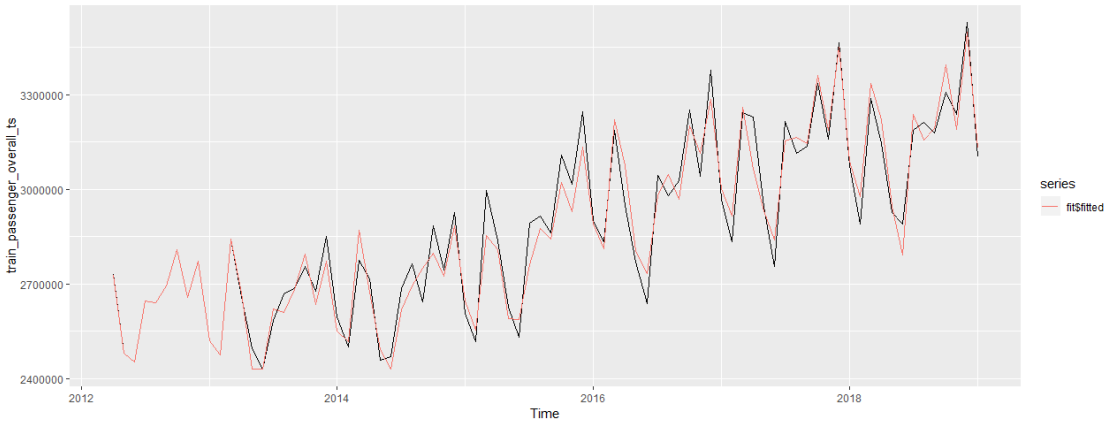


Figure 4.4: ARIMA$(0,1,1)(0,1,1)_{12}$ model fitted on air passenger data

is 59 604 for this model. This also points to a good model considering that the average number of passengers per month stands at close to three million.

Figure 4.4 shows a plot of the fitted ARIMA$(0,1,1)(0,1,1)_{12}$ model for air passengers and the actual observations. The model produced satisfactory results on the training data.

The residuals of the fitted ARIMA$(0,1,1)(0,1,1)_{12}$ model are shown in Figure 4.5. The ACF plot shows no significant spikes, indicating that the error series is uncorrelated and represents a white noise series.

The time plot of the residuals shows that the variation remains the same over time, except for the first few observations with exactly zero variation. The plot also shows that all the information in the data is accounted for as there are no notable patterns in the errors. The histogram of residuals suggests that the residuals are normally distributed with a mean close to zero.

Formal tests can also be used to investigate the presence of autocorrelation. These tests are called portmanteau tests. Instead of looking at individual autocorrelations, a portmanteau test looks at a group of autocorrelation (Hyndman and Athanasopoulos, 2018). One such test is the Ljung-Box test. The value of the Ljung-Box test (Q^*) can be calculated as

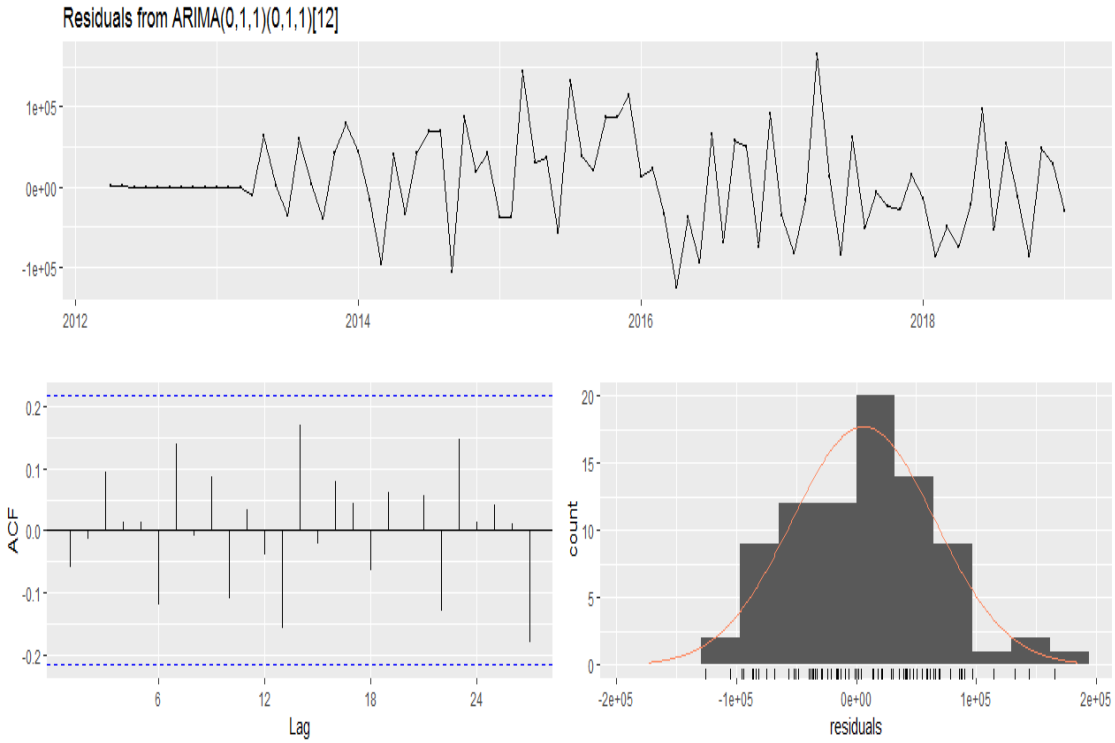


Figure 4.5: Residuals of the fitted ARIMA$(0,1,1)(0,1,1)_{12}$ model

$$Q^* = T(T+2)\sum_{k=1}^{h}(T-k)^{-1}r_k^2, \tag{4.2}$$

where T is the number of observations, h is the maximum lag being considered, r_k is the autocorrelation for lag k and k is the number of parameters in the model. Large values of Q^* indicate that the autocorrelations do not come from an uncorrelated series. The function *checkresiduals()* from the *forecast* package in R can be used for the test and can also be used to produce residual diagnostic charts.

For the selected ARIMA model, the value of Q^* is calculated as 12,38 with a p-value of 0,5754. This is an indication that the autocorrelations are no different from a white noise series.

The values of RMSE and MAE on air passenger time series increased from 59 604 and 46 274 on the training data to 91 643 and 81 043 on the test data. The value of MAPE also increased from 1,6% to 2,5% on the test data, while the values of MASE and Theil's

U statistic remained below one. A plot of the forecasted values from the model is also shown in Figure 4.6. The forecasted values generally match those from the actual observations, although there is evidence of higher residuals at the peaks and lows of the

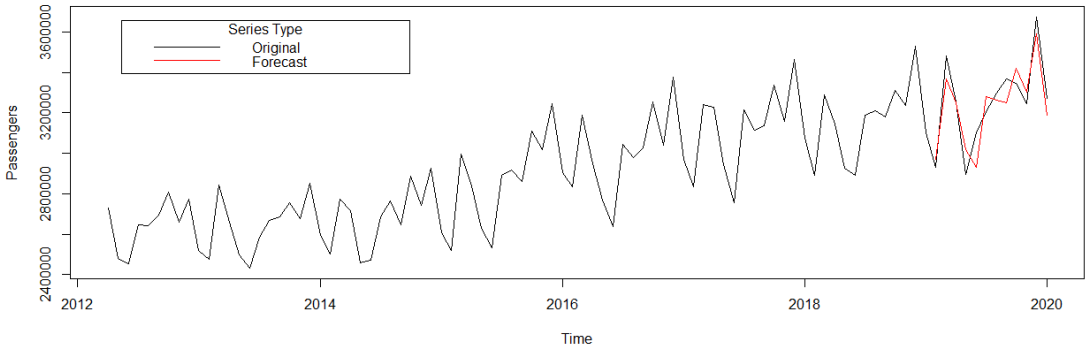


Figure 4.6: Forecasts of ARIMA$(0,1,1)(0,1,1)_{12}$ model for air passengers on the test set

time series compared to other observations. This is clearer in forecasts for March and December 2019. These observations were all underestimated which is an indication that the model can still be improved.

4.6 Fitting the Artificial Neural Networks Model

The *nnetar()* function in the R *forecast* package implements a feed-forward neural network with a single hidden layer and lagged inputs for forecasting univariate time series (Hyndman et al., 2018). This function allows for an automatic model selection of an ANN model and was used in this section to fit ANN models for the time series data under investigation. An ANN model can be written as NNAR$(p,P,k)_{[m]}$, where k is the number of hidden nodes, the parameter p is the number of lagged inputs and P is the number of seasonal lags. The parameters P and p are equivalent to the order of the seasonal AR and the order of non-seasonal AR in the ARIMA model respectively and m is the length of the season.

An NNAR$(5,1,4)_{[12]}$ model was identified for the passenger data. This model contains a network with five input nodes, four hidden nodes and one lagged seasonal input.

Figure 4.7 shows a plot of the fitted ANN model compared to actual observations, which suggests that the ANN model is of acceptable fit. Table 4.3 shows the performance

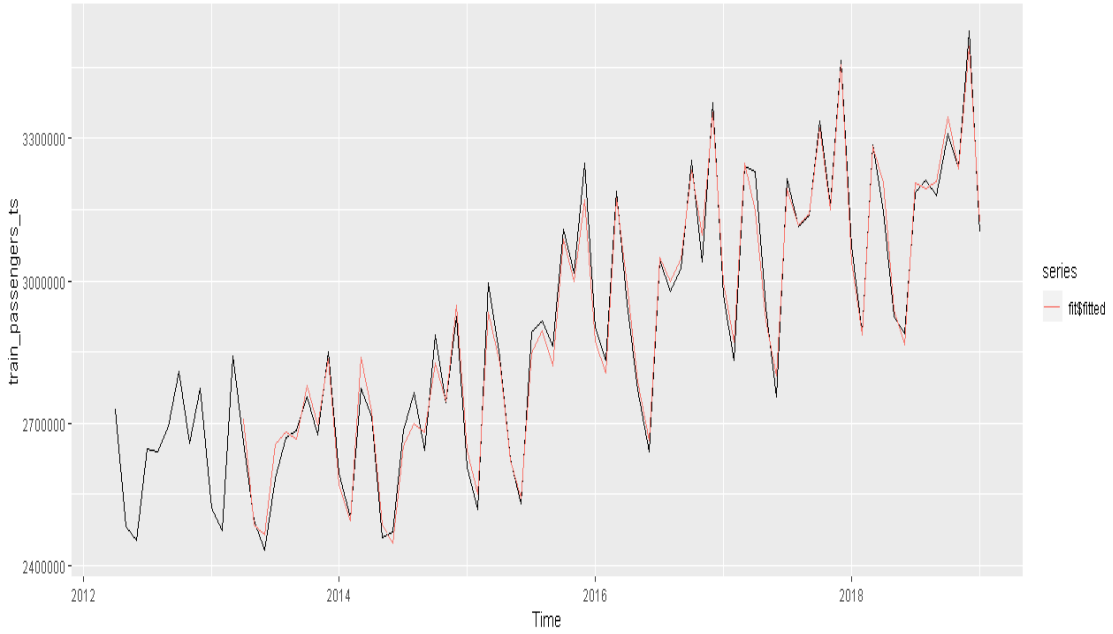


Figure 4.7: NNAR$(5,1,4)_{[12]}$ model fitted on air passenger data

measures of the fitted ANN model. The value of MAPE on the training set is 0,936, which means that, on average, the model can be expected to produce forecasts that are within 0,9% of the actual observations. Both the values of Theil's U statistic and MASE are below one.

Table 4.3: Performance measures of the fitted ANN model

Data	**Data type**	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
Air Passengers	Training set	−37,640	33034,94	27062,85	−0.029	0,936	0,263	0,140
	Test set	59915,340	96902,880	76799,790	1,757	2,299	0,745	0,355

Figure 4.8 shows the residual analysis of the ANN model fitted to the data. The time plot of the residuals shows no clear pattern and the errors are oscillating above and below zero. The time plot also shows that the variance of the residuals remains stable over time. This is also evident from the ACF plot, as there is only one significant autocorrelation at lag 12.

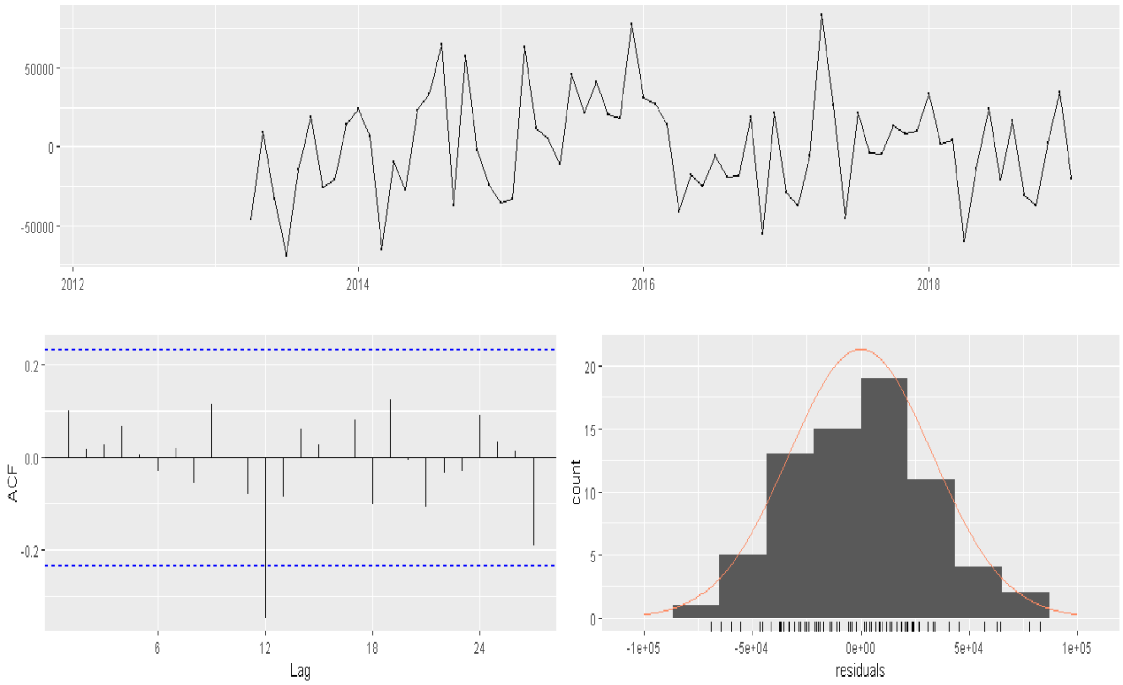


Figure 4.8: Residuals for the fitted ANN model for air passengers

The histogram of the residuals suggests that the distribution of the series is symmetric with a mean close to zero and a constant variance. The Ljung-Box test statistic is 16,323 with 18 degrees of freedom. This produced a p-value of 0,57, which indicates that the error series is equivalent to a white noise series, in turn showing that the model is suitable for forecasting values for this data set.

The value of the RMSE jumped from 33 034 on the training set to 96 902 on the test set. This degrading performance can also be observed on MAE which increased from 27 062 to 76 799 on the test set. However, the Theil's U statistic and MASE still showed values that are below one. A plot showing the model performance on the test set is shown in Figure 4.9.

The plot shows that the pattern of forecasted values generally follows that of the actual observations. A closer look at the peaks reveals that the model also suffers from the underestimation observed in the ARIMA model. The absolute residuals in this case are higher compared to the ARIMA model.

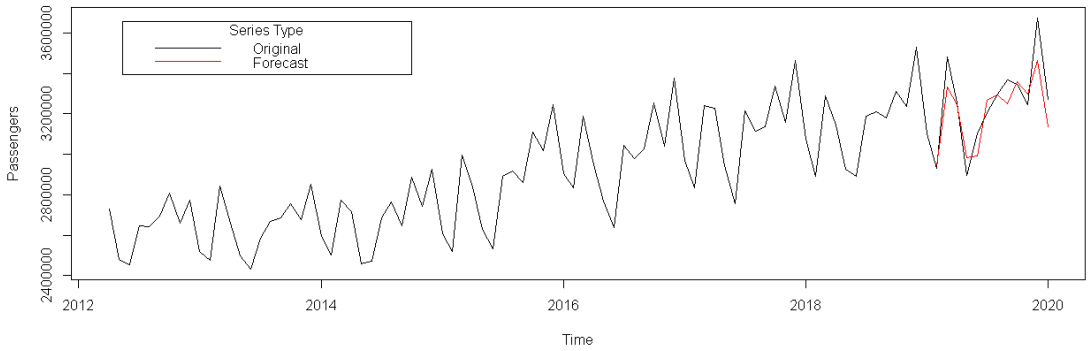


Figure 4.9: Forecasts of ANN model for air passengers

4.7 Fitting the Support Vector Machines Model

In fitting the SVM model, the *e1071* package in R written by Meyer et al. (2018) was used to fit the time series for the passengers. The function *tune.svm()* allows for the fitting of a specified SVM model as discussed in Section 2.4. The function also provides the ability to automatically tune the model based on the intervals provided for the different parameters of the SVM model. The parameters that need tuning are Epsilon, Cost and Gamma. For this modelling approach, the monthly number of passengers is represented by y while x represents the time.

The automatically tuned model parameters are 1, 0,08 and 0,1 for the Cost, Gamma and Epsilon respectively. The results of the fitted model are shown in Figure 4.10. Visually, the model seems to miss the seasonality of the time series and instead chose a straight line as a better fit for the time series.

Table 4.4 shows the performance measures of the fitted SVM model. The value of MAPE is 4,547 and the value of MAE is 129 319,5. The values of MASE and Theil's U statistic are both below one.

Table 4.4: Performance measures of the fitted SVM model

Model	**Data type**	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
SVM	Training set	−15092,27	165949,2	129319,5	−0,898	4,547	0,707	0,764
	Test set	98522,52	126370,7	103313,6	2,958	3,124	0,444	0,462

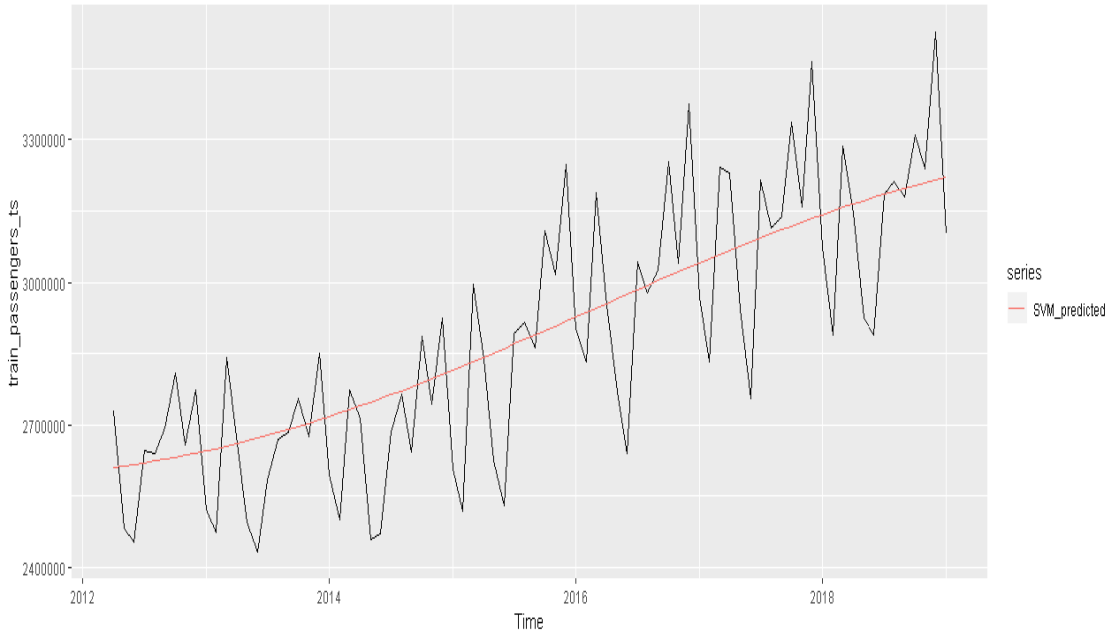


Figure 4.10: Fit of the SVM model for air passengers

Figure 4.11 shows the residual analysis of the SVM model fitted to air passenger data. The time plot of the residuals shows an observable pattern with a mean close to zero. The ACF plot of the errors shows three significant spikes at lags 6, 12 and 18. This is an indication that the model has not accounted for some of the variations in the time series. The huge spike at lag 12 shows that seasonality was also not accounted for in the model. This is not surprising given the observations in Figure 4.10. The histogram of the errors shows that the majority of the observations are in the middle.

In their article titled "Time series forecasting by a seasonal support vector regression model", Pai et al. (2010) suggested a method for forecasting time series data with a seasonal component. The method is called seasonal support vector regression. Their modelling procedure begins by splitting the time series into its basic components. The time series is then deseasonalised and an SVM model is fitted to the deseasonalised time series. In order to select optimal values for the parameters, a hybrid genetic algorithm and tabu search called GA/TS is used. Finally, the forecasts are produced by combining the forecasts from the SVM with the seasonal estimates from the decomposition method.

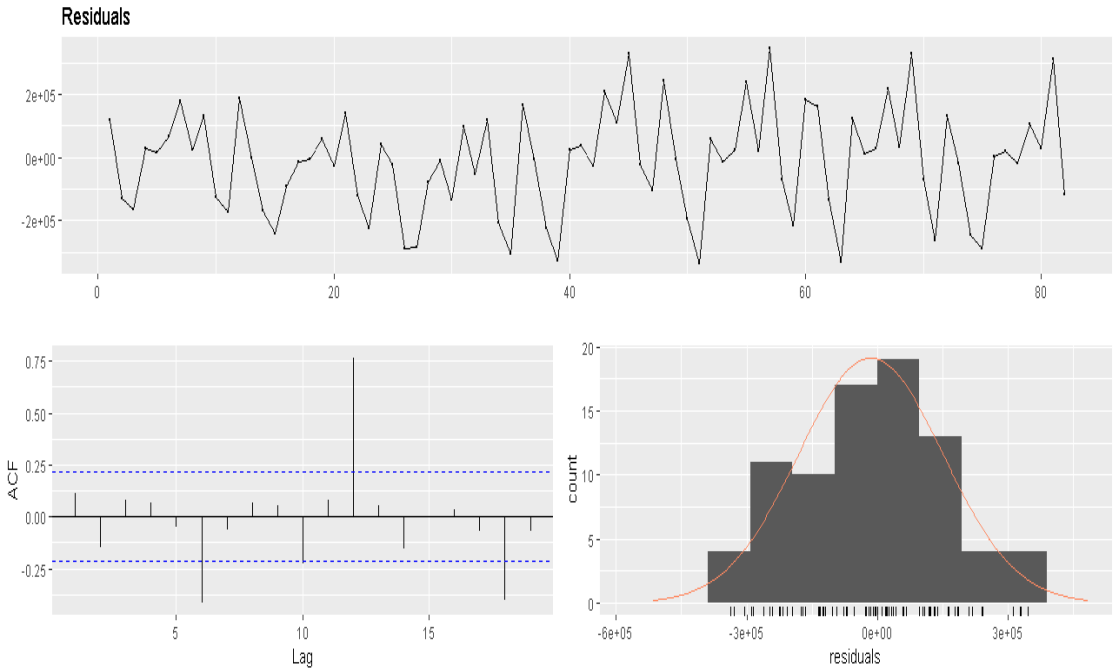


Figure 4.11: Residuals of the fitted SVM model

In order to account for seasonality in the SVM model fitted to air passengers, a similar methodology is followed. However, instead of using the GA/TS method for fitting the SVM, the *e1071* package was used to fit the deseasonalised time series. The procedure is shown in Figure 4.12.

The air passengers time series was decomposed by means of seasonal and trend decomposition using Loess (STL). The R function *mstl()* was used to decompose the time series. Using the *tune.svm()*, the values of the parameters were calculated as 1, 10 and 0,1 for Cost, Gamma and Epsilon respectively. Table 4.5 shows the performance of the model on the training data.

Table 4.5: Performance measures of the SVM model for deseasonalised air passengers

Model	**Data type**	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
Seasonal SVM	Training set	333,694	46472,39	38307,37	−0,014	1,338	0,696	0,743
	Test set	78768,87	130327	104566,9	2,294	3,174	0,450	0.468

The fitted model showed significant improvements on the training set. The value of MAPE has dropped from 4,55 to 1,34 compared to the original model. All the other

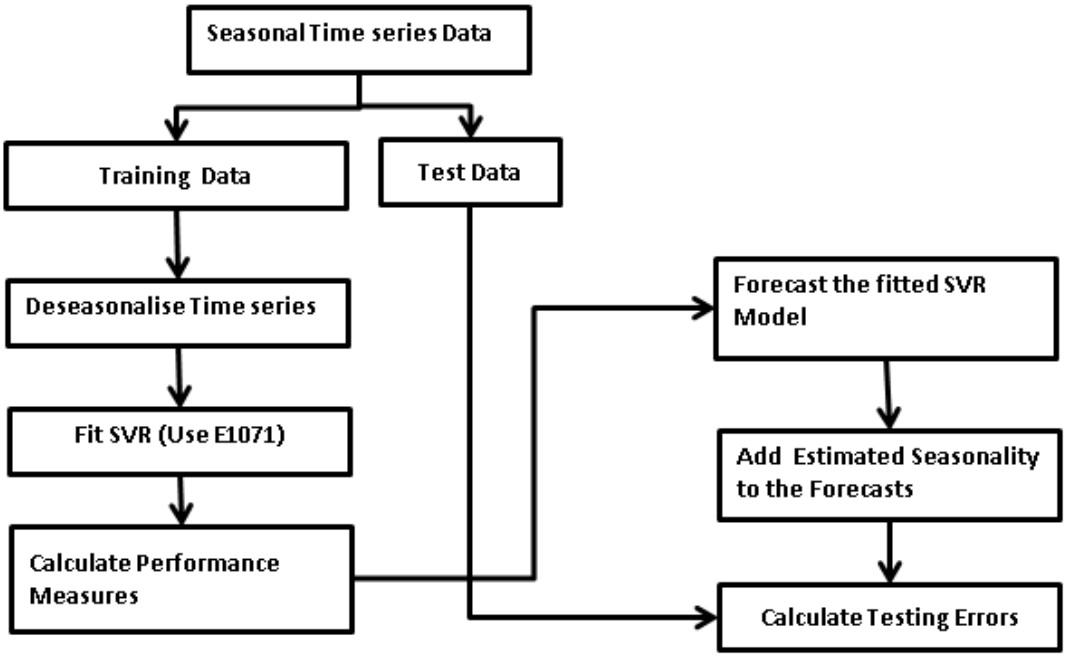


Figure 4.12: SVR accounted for seasonality

measures also showed significant improvement. Figure 4.13 shows the residuals from the fitted model.

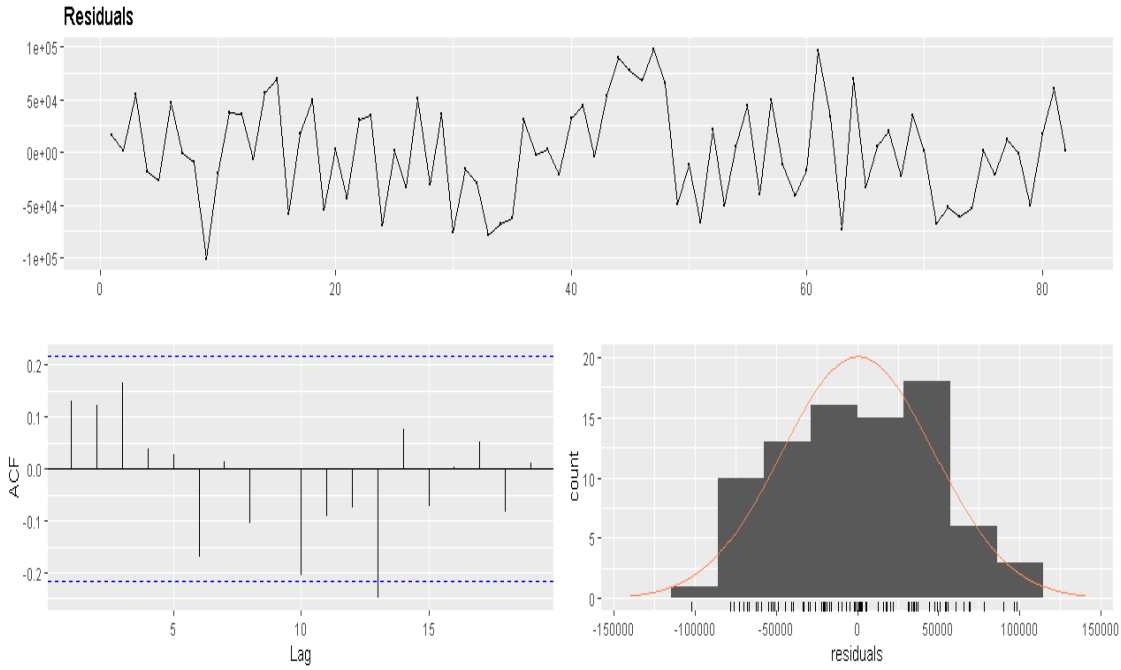


Figure 4.13: Residuals of SVR for deseasonalised air passenger data

The huge spike at lag 12 has disappeared and the residual plot also shows no observable pattern. This shows that seasonality was successfully removed from the data. After a suitable model was fitted to the deseasonalised time series, the seasonal component was added back to the fitted model for comparison with the training data. Figure 4.14 shows the fitted model compared to the training data. The results show that the model is able to track the changes in the level of the series.

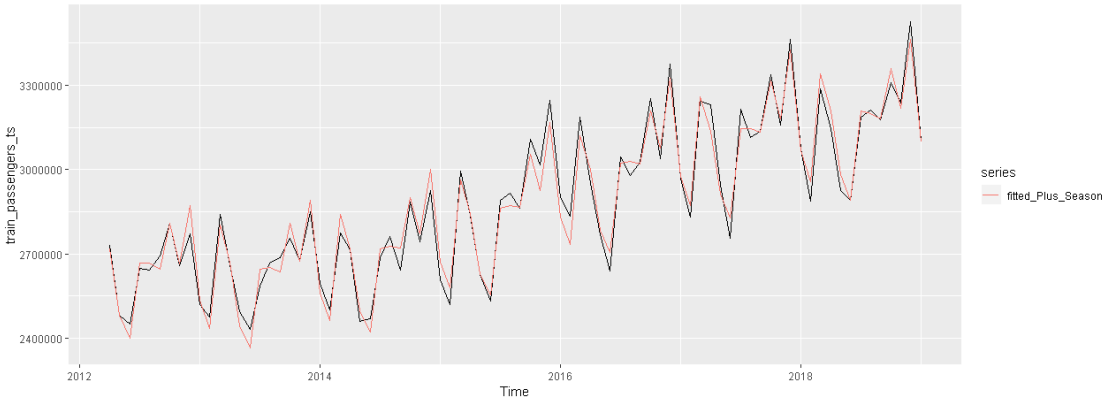


Figure 4.14: Fitted SVR with seasonality added

The performance measures on the test data show improved results for the fitted SVM model applied to the original time series. The value of the MAE has improved by 15,6% to 103 313 for the test set compared to the training set. The value of RMSE has also improved from 157 772 to 126 371 on the test set. There is also a clear improvement in the value of the MAPE, which decreased from 4% to 3%, an improvement of one percentage point. The values of the MASE and Theil's U statistic remained within the acceptable level of below one.

The model with deseasonalised time series produced a MAPE of 3,17, which is more than twice the performance compared to the training set. The model also failed to perform better than the model fitted to the original time series based on the values of MAE and RMSE, although the results were close. However, the model managed to perform well on MPE and ME. A plot of the forecasts from the two models is also shown in Figure 4.15.

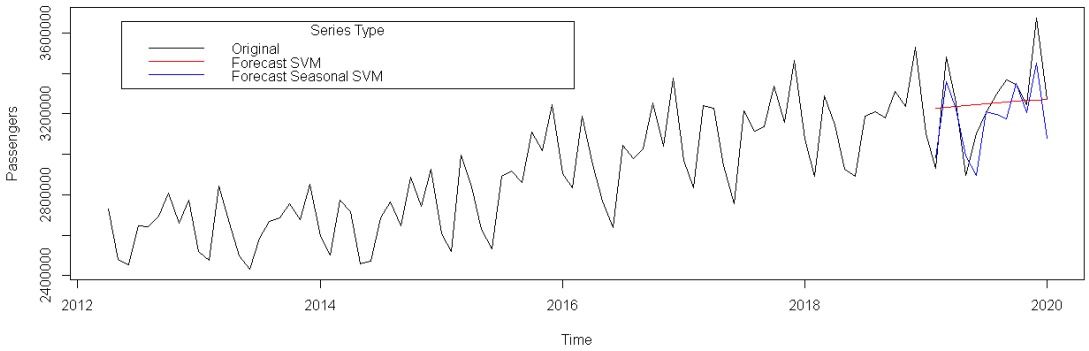


Figure 4.15: Forecasts of the SVM models for air passengers on test set

The figure shows that the seasonal model is able to predict the actual observations of the test set. However, the plot also shows deteriorating performance as the forecast window increases. On the other hand, the model fitted to the original series shows consistency as the forecasting time horizon increases.

4.8 Fitting the Regression Model With ARIMA Errors

To fit a regression model with ARIMA errors, the same function used for fitting a normal ARIMA model was used. In addition, an argument *xreg* was used to allow for the fitting of the regression part of the model. The argument takes on a vector of inputs equal in size to the time series under investigation (Hyndman and Athanasopoulos, 2018).

A vector of length 94 was generated, which should be of the same size as the time series under investigation. This vector holds a sequence of numbers starting from 1 to 94 and was converted to a time series starting from April 2012 to January 2020. This new time series was used in the argument *xreg* of the function for the time series under investigation.

A regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors was identified with parameters as shown in Table 4.6. This model indicates that the error series is seasonal since there is a seasonal part in this model, however, there is no any form of differencing in this model. The model has an MA part of order one and a seasonal AR part of order one.

Table 4.6: Coefficients of the regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors

Parameter	Coefficient	Standard error
MA(1)	0,4125	0,0944
Seasonal_AR(1)	0,8687	0,0413
Intercept	2589233,25	68812,77
Slope	7366,1310	986,4965

Using the data in Table 4.6, the equation of regression with ARIMA errors can be written as

$$Y_t = 2589233{,}25 + 7366{,}13X_t + N_t. \tag{4.3}$$

N_t is a remainder from the regression model and it is modelled as follows:

$$N_t = 0{,}87N_{t-12} + e_t - 0{,}41e_{t-1}, \tag{4.4}$$

where e_t represents the residuals of the model. Figure 4.16 shows a plot of the fitted regression model with ARIMA errors. The fitted model shows an improvement as the forecasting period moves from the left to the right. In the period between 2012 and 2015, the fitted model seems to underestimate the lowest values in the time series.

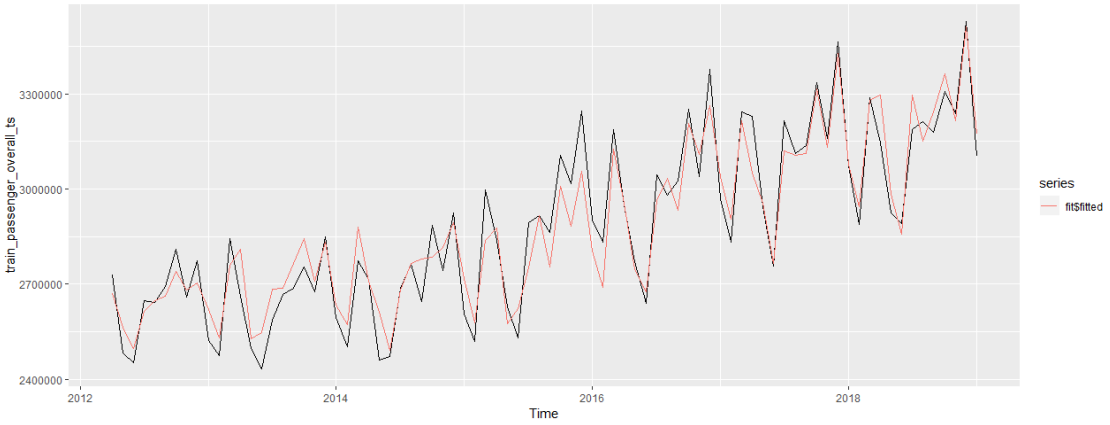


Figure 4.16: Fit for the regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors

Table 4.7 shows the performance measures of the fitted model. The value of MAPE on the training set is 2,280, while the values of MASE and Theil's U statistic are 0,632 and 0,367 respectively. The errors of the model are also expected to be around 65 069 as

measured by the value of MAE.

Table 4.7: Performance measures of the fitted regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors

Passenger Type	**Data type**	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
All	Training set	−1457,905	79724,41	65069,17	−0,168	2,280	0,632	0,367
	Test set	13201,449	83800,70	73454,36	0,296	2,266	0,713	0,303

Figure 4.17 shows the residual analysis of the fitted model. The time plot of the residuals shows a clear pattern, which suggests that there is some information in the residuals that was not accounted for in the model. This is also evident from the ACF plot as there are a number of significant autocorrelations.

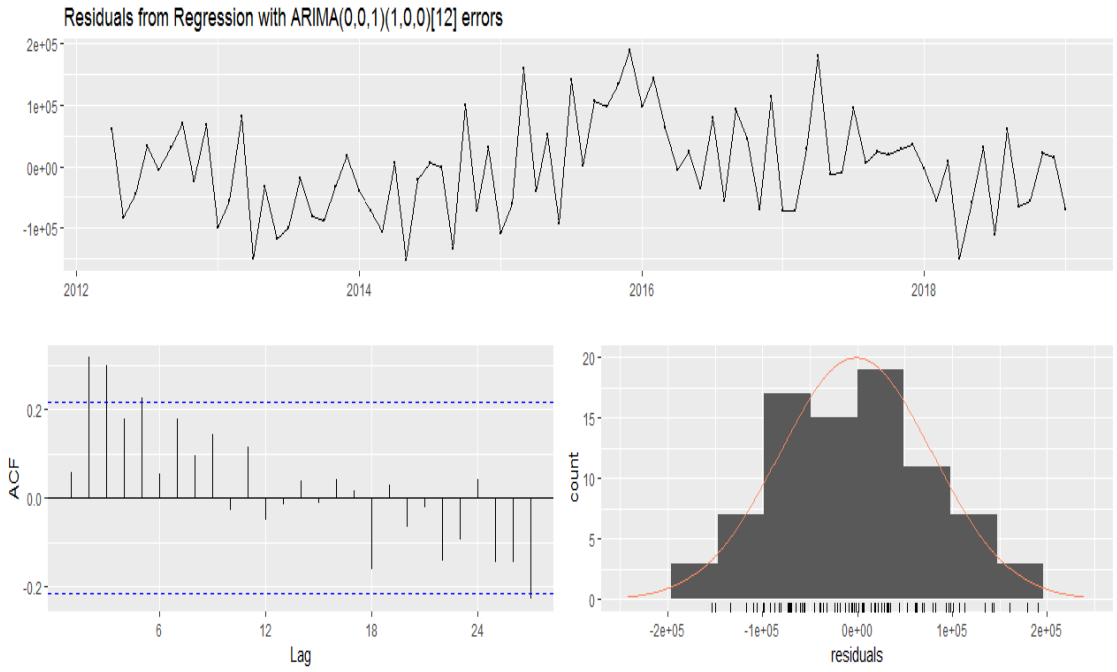


Figure 4.17: Residuals for the fitted regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors

The histogram of the residuals suggests that the distribution of the series is not symmetric. The plot also suggests that the residuals have a mean close to zero and a constant variance. In addition, the Ljung-Box test indicates that the residual series, with a significant p-value of 0,001359, is not white noise.

The performance measures on the test data show results that are close to those obtained from the training set. The only exception is on the ME which jumped from $-1\,458$ on

the training set to 13 201 on the test set. The values of the MASE and Theil's U statistic also remained within the acceptable levels of below one. A plot of forecasts from the model is also shown in Figure 4.18.

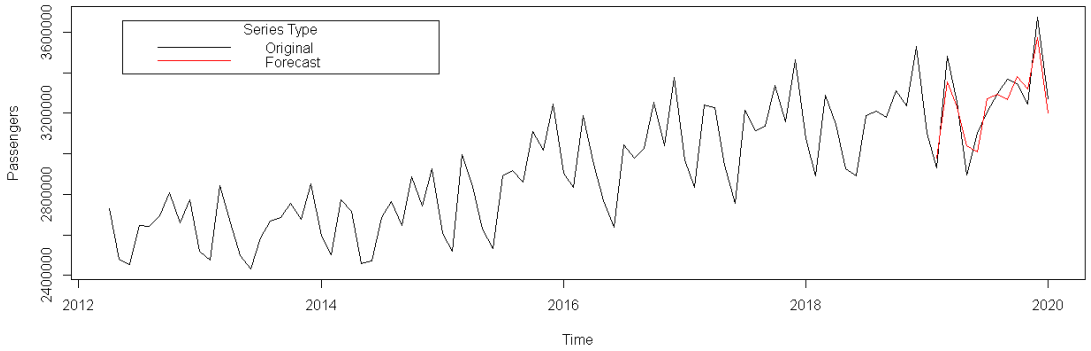


Figure 4.18: Forecasts of regression model with ARIMA errors for air passengers on the test set

The plot shows that the regression model with ARIMA errors also indicates significant gaps that appear at the peaks of the time series. These gaps, however, appear smaller when compared to the other models considered.

4.9 Comparing the Performance of the Methods

Table 4.8 shows the performance of the different models on both the training set and test set.

Table 4.8: Performance measures of the fitted models on all data sets

Method	Data type	ME	RMSE	MAE	MPE	MAPE	MASE	Theil's U
ARIMA	Training set	5957,163	59604,47	46273,640	0,184	1,585	0,450	0,273
	Test set	21071,795	91643,330	81043,460	0,566	2,503	0,787	0,338
ANN	Training set	−37,640	33034,94	27062,85	−0.029	0,936	0,263	0,140
	Test set	59915,340	96902,880	76799,790	1,757	2,299	0,745	0,355
SVM	Training set	40732,09	157772,9	122457,3	1,167	4,312	0,670	0,737
	Test set	98522,52	126370,7	103313,6	2,958	3,124	0,444	0,462
Seasonal SVM	Training set	333,694	46472,39	38307,37	−0,014	1,338	0,696	0,743
	Test set	78768,87	130327	104566,9	2,294	3,174	0,450	0.468
Regression with ARIMA errors	Training set	−1457,905	79724,41	65069,17	−0,168	2,280	0,632	0,367
	Test set	13201,449	83800,70	73454,36	0,296	2,266	0,713	0,303

Based on the value of RMSE, the selected ANN model produced the best value of 33 035 for the training set. This model was followed by the seasonal SVM model with an RMSE

value of 46 472. The ARIMA$(0,1,1)(0,1,1)_{12}$ model followed with an RMSE value nearly twice the best value at 59 604. The original SVM model came in last with an RMSE value of 157 773. This is more than four times the RMSE value of the best performing model. The ordering of the models based on the best RMSE is the ANN, seasonal SVM, ARIMA$(0,1,1)(0,1,1)_{12}$, regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors and SVM. The same performance and ordering can also be observed for MAE. The absolute errors remained high for SVM and low for the ANN model.

Looking at MAPE, the chosen ANN model achieved a value below a percentage point on the training set, followed by the seasonal SVM model with MAPE of 1,34%. All the models produced values of MAPE that are below 5%. Coupled with the values of MASE and Theil's U statistic, which are below one, this is an indication that the four methods have performed well on the training set of air passenger data.

Figure 4.19 shows the forecasts from the different methods and the test data. All the

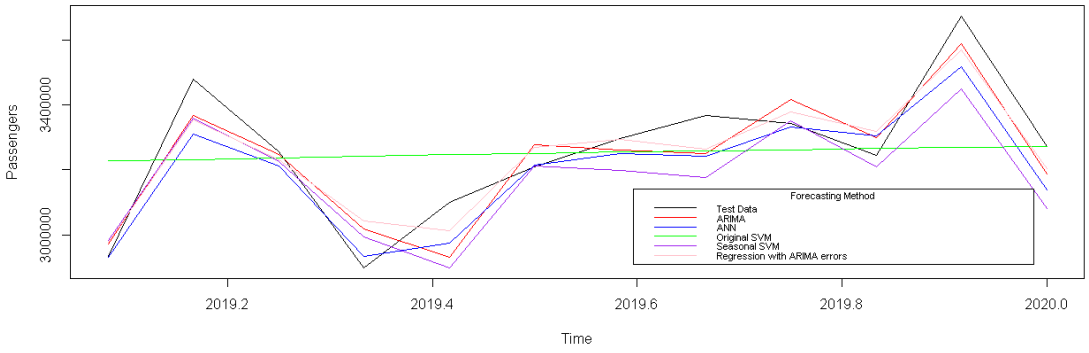


Figure 4.19: Combined forecasts of the different methods and the test set

models produced acceptable results based on the values of the MASE and Theil's U statistic. These values are way below one for all the models, giving an indication that all the models are better than a naïve model which is a benchmark model for forecasting models.

Looking at the RMSE, the regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors produced the best value for the RMSE at 83 801 compared to the other models. This was

followed by the ARIMA$(0,1,1)(0,1,1)_{12}$ model with an RMSE value of 91 643. Similar to the training set, the original SVM produced the worst results on the test set compared to the other methods with an RMSE value of 126 371 for the model fitted to the original data and 130 327 on the seasonal model.

The performance of the models when compared using the value of the MAE shows that the regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors remained the better method with a MAE of 73 454. This regression model was followed closely by the ANN method, with the chosen model producing a MAE of 76 800. Again, the chosen SVM model produced the worst results with a MAE of 103 314.

The regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors remained the best model on all the measures considered, producing the best MAPE of 2,266 compared to the other models. This model was closely followed by the ANN model with a MAPE of 2,299. The SVM performed the worst on all the measures considered here, even though the model performance improved on the test set compared to the training set.

Although the worst performing model, the selected SVM model is the only model that showed improved results on the test set when compared to the training set. All other models showed degrading performance on the test set. For example, the selected ANN model performed better on the test set but could not replicate this performance on the test set.

4.10 Bagging the Models

In this section, the models are bagged to see if they can improve their performance on the test set. The bagging method used here was explained by Bergmeir, Hyndman, and Benítez (2016) and is included in the *forecast* package. Firstly, 10 different time series were created by using a bootstrap method on the training set. Figure 4.20 shows the bootstrapped time series for the air passenger data. It can be observed from the plot that the bootstrapped data show increased variability in the months of July and August.

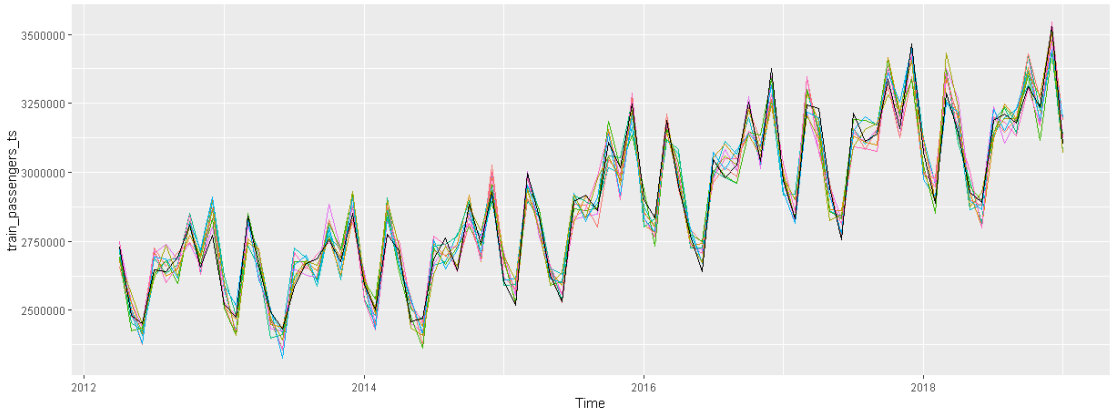


Figure 4.20: Bootstrapped air passenger data

Secondly, a forecasting model was fitted to each time series for a given forecasting method and grouped by forecasting method, for example ARIMA. Forecasts were then produced for each time series using the fitted model. Finally, the mean of the forecasts for each month was taken as the forecast for the month. This new time series, consisting of mean forecasts, was compared with the test set. Table 4.9 shows the performance of the bagged models on the test set.

Table 4.9: Performance measures of the fitted bagged model on test data

Model	ME	RMSE	MAE	MPE	MAPE
Bagged ARIMA	92,573	89638,36	81783,8	−0,096	2,537
Bagged ANN	44257,04	94523,97	78785,18	1,262	2,384
Bagged SVM	50870,19	212405	165450,7	1,164	5,060
Bagged seasonal SVM	86413,9	134573	108178,4	2,527	3,272
Bagged Regression model with ARIMA errors	2664,602	89255,48	80148,47	−0,036	2,476

The bagged ARIMA model showed improvements on both the RMSE and MPE compared to the ARIMA model without bagging. Thus, the use of a bagged model improved the value of the RMSE by over 2% from 91 643 to 89 638. The MPE also improved from 0,566 to −0,096. On the other hand, the values of the MAPE and MAE did not improve, in fact they deteriorated. The value of the MAPE increased from 2,5% to 2,54% while the value of the MAE increased from 81 043 to 81 784 passengers. The forecasts from the ARIMA and the bagged ARIMA models on the test set are shown in Figure 4.21(a).

The bagged ANN model also showed improved performance in the RMSE and MPE. The value of the RMSE decreased from 96 903 to 94 524 compared to the originally fitted ANN model. This is an improvement of 2,45%. The value of the MPE also improved from 1,76 to 1,26 on the bagged ANN model. Similar to the bagged ARIMA model, the values of the MAE and MAPE did not improve for the bagged ANN model. Figure 4.21(b) shows the bagged ANN forecasts and ANN models compared to the test set.

The bagged SVM model applied to the original time series deteriorated in performance. The value of the RMSE increased from 126 371 to 212 405, an increase of 68% compared to the original model. There was a similar deterioration in performance on the MAE, which increased by 62 135 to 165 451. The value of the MAPE also increased from 3,12 to 5,06, an increase of 1,94. However, there were some improvements from the MPE and the ME. The value of the MPE improved from 2,96 in the original model to 1,16 in the bagged model. Figure 4.21(c) shows a plot of the forecasts of the original model against the bagged model.

The bagged seasonal SVM model also showed no improvement when compared to the original seasonal SVM model. The value of the RMSE increased from 130 327 to 134 573 and there was also a slight increase in the value of the MAE from 104 567 to 108 178 as well as a slight increase the MAPE from 3,17 to 3,27. Figure 4.21(d) shows forecasts of both the bagged model and the original model on the test set.

There was also no improvement in performance on the bagged regression model with ARIMA errors. The value of the MAPE increased from 2,27 to 2,48, as did the values of the RMSE and the MAE. The only measure that produced improved results was the MPE; the absolute value of the MPE improved from 0,296 to 0,036 on the bagged model. Figure 4.21(e) shows forecasts of the original model against the bagged model.

A comparison of the bagged models with regard to the MAPE shows that the bagged ANN model performed better than the other bagged models with a MAPE of 2,38. This was followed by the bagged regression model with ARIMA errors which had a MAPE

of 2,47. The bagged model that performed the worst on the value of the MAPE was the SVM model. This model produced a MAPE of 5,06. Interestingly, all the models still performed worse than their original counterparts on this measure of performance. The same order of performance was also observed for the MAE. The bagged ANN model produced the lowest MAE of 78 785.

On the values of the RMSE, the bagged regression model with ARIMA errors performed better than the other bagged models with an RMSE value of 89 255. This was closely followed by the bagged ARIMA model which produced an RMSE value of 89 638. The bagged regression model with ARIMA errors also produced the best MPE absolute value of 0,036 followed again by the bagged ARIMA model with an MPE absolute value of 0,096.

When measuring the performance of the bagged models, mixed results were found. The values of the RMSE and MPE improved for the bagged ARIMA and ANN models, while the other measures did not when compared to the original models. These slight improvements did not, however, affect the outcome of the analysis made using the original models.

4.11 Summary

This chapter introduced an air passenger data set containing information of passengers from April 2012 to January 2020. The information pertained to passengers passing through South African airports and was disaggregated by the type of passenger. The types of passenger included international, regional, domestic and unscheduled. The data set was converted to a time series and further split into a training set and a test set. The data from the period April 2012 to January 2019 were used for the training set and the remainder were used as the test.

The fitted models were explored and compared using forecast accuracy measures such as the RMSE, MPE, MAE, MAPE and Theil's U statistic. The Ljung-Box test was also conducted to see if the residuals of the fitted models were a white noise series. All the methods considered in this study performed very well on the time series analysed, giving values for MASE and Theil's U statistic of less than one.

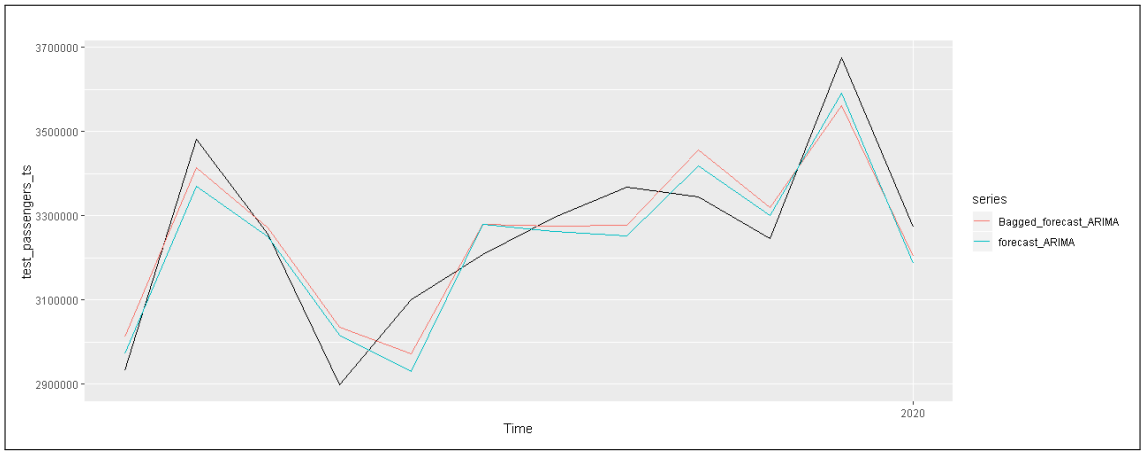
The artificial neural network NNAR$(5,1,4)_{12}$ model outperformed the other models on the training data, performing better than the other models on most of the accuracy measures used. This model was followed by ARIMA$(0,1,1)(0,1,1)_{12}$. The regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors came out third overall.

Based on the performance of the models on the test set, the regression model with ARIMA$(0,0,1)(1,0,0)_{12}$ errors performed better than the other methods on the air passenger data. This is an interesting observation considering that the residuals from this model failed the Ljung-Box test, which was an indication that the model did not capture all the available information. The ARIMA$(0,1,1)(0,1,1)_{12}$ model again came second, which shows the consistency of the ARIMA method in general.

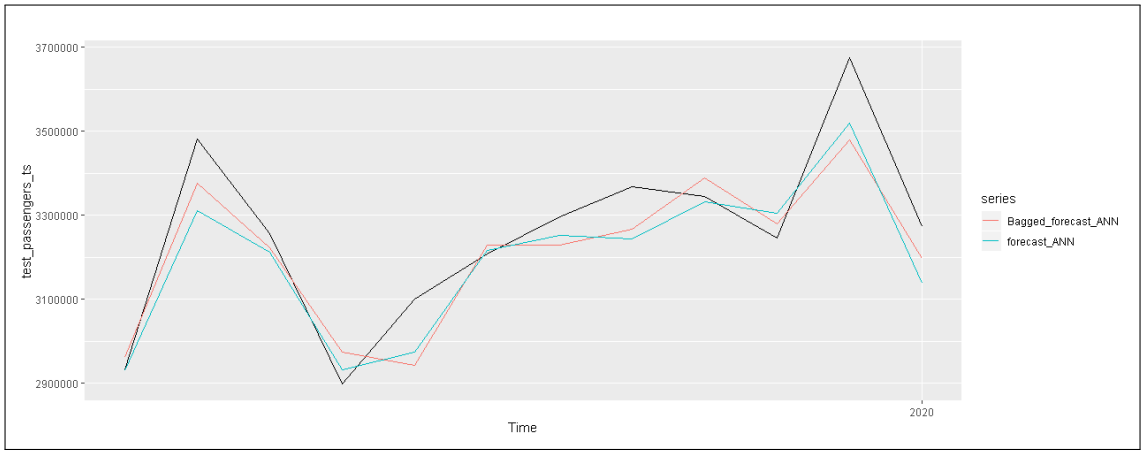
The SVM models performed worst on this data set according to the accuracy measures. This is despite having improved their performance compared to the training set. In fact, the SVM models were the only models that showed improved performance on the test set. Despite coming out first on the training data, the selected ANN model failed to emulate that performance on the test set. Consequently, the SVM and ANN models

failed to perform better than the regression with ARIMA errors and the ARIMA models. The ANN and SVM are traditionally machine learning methods. These findings are in line with those made by Makridakis, Spiliotis, and Assimakopoulos (2018), who found that the machine learning methods are dominated by the traditional statistical methods.

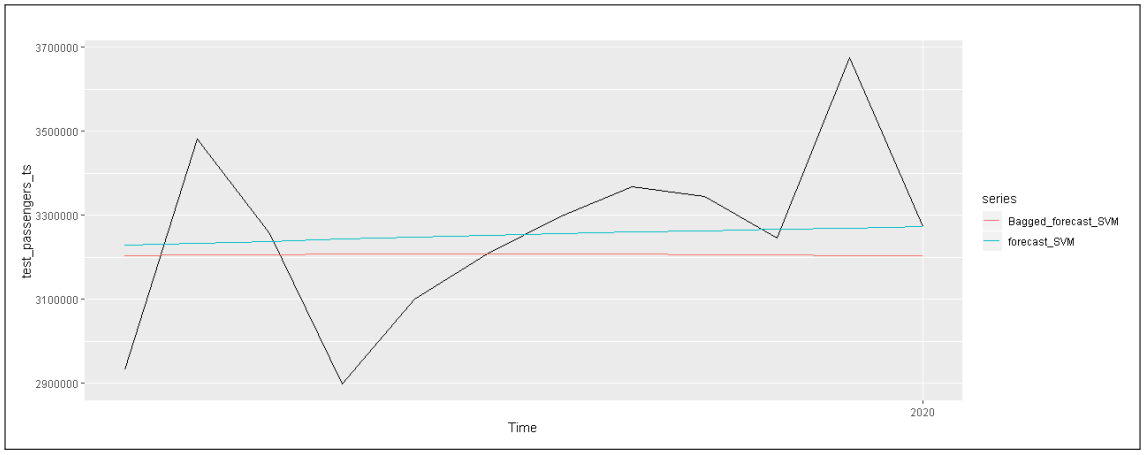
A bagging procedure using a moving block bootstrap was also applied to the time series in order improve the performance of the different forecasting methods. The bagged models showed mixed results when compared with the original models. There was an improvement in the performance of the bagged ANN and ARIMA models when compared on the values of RMSE and MPE, but these results did not influence the order of performance observed with the original models. It is therefore evident that bagging helps reduce bias in a forecasting model.



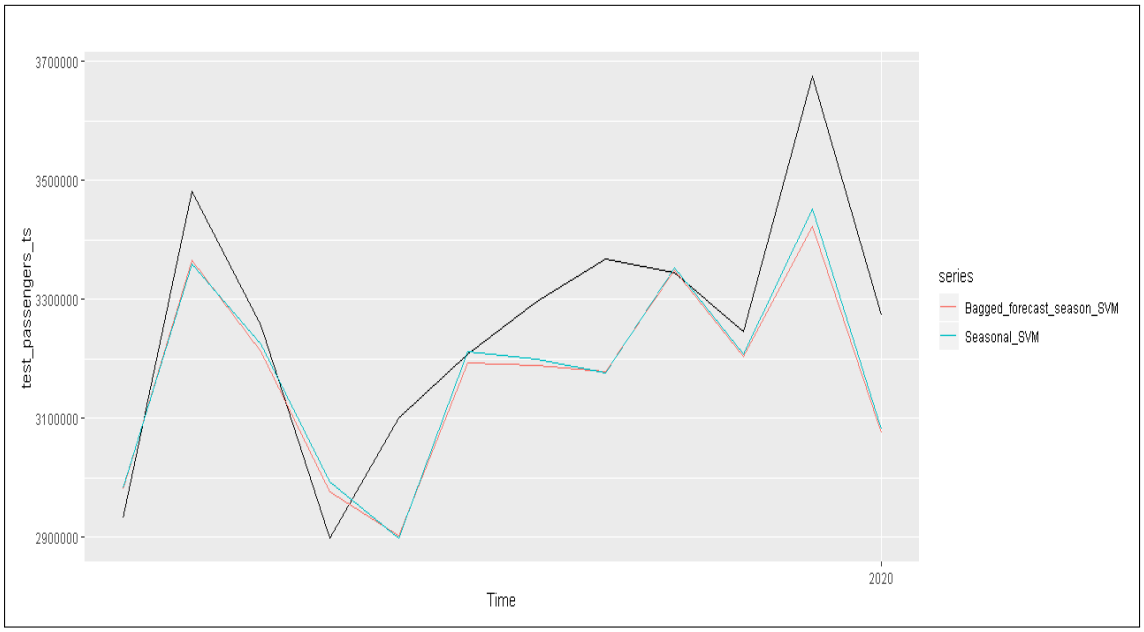
(a) ARIMA forecasts



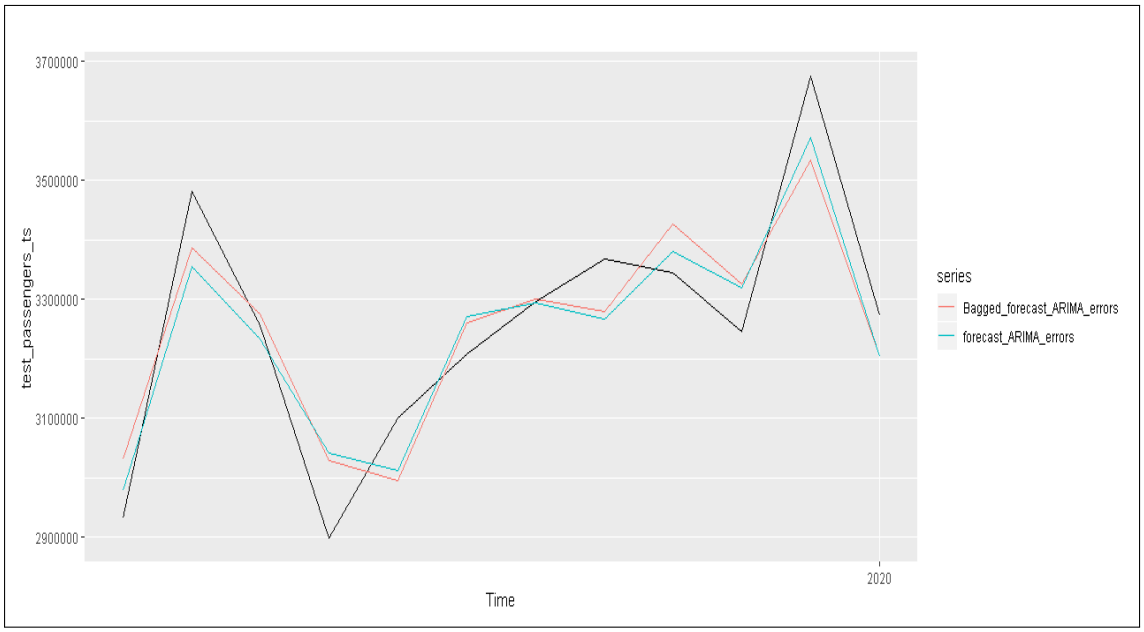
(b) ANN forecasts



(c) SVM forecasts



(d) Seasonal SVM forecasts



(e) Regression model with ARIMA errors forecasts

Figure 4.21: Forecasts from the original and bootstrapped air passenger data

Chapter 5

Conclusions and Recommendations

The study was aimed at studying advanced time series forecasting methods. The methods were also tested using monthly South African air passenger data. It can be concluded that the traditional statistical forecasting methods (ARIMA and regression model with ARIMA errors) performed better than the machine learning methods (ANN and SVM) on this data set, based on the measures of accuracy used. This is in line with a study conducted by Makridakis, Spiliotis, and Assimakopoulos (2018). This research used aviation data for illustration, but more data sets could be added for a wider comparison. The application of the methods to different data sets might result in the other methods performing better.

The machine learning methods performed well on the training data but could not maintain the same performance on the unseen data, with their performance deteriorating substantially when compared to the training data. The traditional statistical methods performed more consistently when comparing their performance on the training data and the test data. In their study, Makridakis, Spiliotis, and Assimakopoulos (2018) also found that the machine learning methods failed to beat the traditional statistical methods during forecasting, despite doing well during the training stage. The reason for this is that these methods are prone to overfitting.

Bagging the methods showed marginal improvement on the SVM and ARIMA methods for the values of MAPE and MPE. The other methods showed no improvement with

some deterioration in performance. It is evident that bagging does not influence the choice of a suitable forecasting method when applied to all the competing methods.

The area of time series forecasting has recently gained traction especially with the introduction of machine learning algorithms for time series forecasting. Current research on time series forecasting includes the development of the necessary theory in the area of machine learning methods. With the increase in the number of forecasting methods, future studies in the area may look at the development of a comprehensive framework that guides the identification of methods to be considered for evaluation. The increase in the number of the ANN variants also makes it difficult to select a specific variant to consider for evaluation.

Bibliography

Adeniran, A. O., Kanyio, O. A., and Owoeye, A. S. (2018). Forecasting methods for domestic air passenger demand in Nigeria. *Journal of Applied Research on Industrial Engineering*. 5.2, pp. 146–155.

Adhikari, R. and Agrawal, R. K. (2013). *An Introductory Study on Time Series Modeling and Forecasting*. URL: `https://arxiv.org/ftp/arxiv/papers/1302/1302.6613.pdf`. Last visited 25 May 2018.

Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews*. 29.5-6, pp. 594–621.

Airports Company South Africa (2020). *Total Passenger Traffic at various Airports Company South Africa airports*. URL: `https://www.airports.co.za/business/statistics/aircraft-and-passenger`. Last visited 25 March 2020.

Aladag, C. and Eğrioğlu, E. (2012). *Advances in Time Series Forecasting*. Bentham eBooks. ISBN: 9781608053735.

Bergmeir, C., Hyndman, R. J., and Benítez, J. M. (2016). Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation. *International Journal of Forecasting*. 32.2, pp. 303 –312.

Breiman, L. (1996). Bagging predictors. *Machine Learning*. 24.2, pp. 123 –140.

Cao, L. and Tay, F. E. (2001). Financial Forecasting Using Support Vector Machines. *Neural Computing & Applications*. 10.2, pp. 184 –192.

Dantas, T. M., Cyrino Oliveira, F. L., and Varela Repolho, H. M. (2017). Air transportation demand forecast through Bagging Holt Winters methods. *Journal of Air Transport Management*. 59, pp. 116 –123.

Eberly College of Science (2019). Applied Time Series Analysis. Available from: `https://newonlinecourses.science.psu.edu/stat510/lesson/2/2.2`.

Fan, Y., Li, P., and Song, Z. (2006). Dynamic Least Squares Support Vector Machine. *2006 6th World Congress on Intelligent Control and Automation*. 1, pp. 4886–4889.

Faraway, J. and Chatfield, C. (1998). Time series forecasting with neural networks: a comparative study using the air line data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*. 47.2, pp. 231–250.

Francis, E., Tay, and Cao, L. (2001). Application of Support Vector Machines in financial time series forecasting. *Omega*. 29.4, pp. 309 –317.

Gibrilla, A., Anornu, G., and Adomako, D. (2018). Trend analysis and ARIMA modelling of recent groundwater levels in the White Volta River basin of Ghana. *Groundwater for Sustainable Development*. 6, pp. 150 –163.

Hamzaçebi, C. (2008). Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*. 178.23, pp. 4550 –4559.

Hikichi, S. E., Salgado, E. G., and Beijo, L. A. (2017). Forecasting number of ISO 14001 certifications in the Americas using ARIMA models. *Journal of Cleaner Production*. 147, pp. 242 –253.

Härdle, W., Horowitz, J., and Kreiss, J.-P. (2003). Bootstrap Methods for Time Series. *International Statistical Review / Revue Internationale de Statistique* [online]. 71.2, pp. 435–459. Available from: `http://www.jstor.org/stable/1403897`.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2016). A Practical Guide to Support Vector Classification. Available from: `http://www.csie.ntu.edu.tw/~cjlin`.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. 2nd ed. OTexts: Melbourne, Australia. Available from: `https://otexts.com/fpp2`.

Hyndman, R. J., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. (2018). *forecast: Forecasting functions for time series and linear models*. Available from: `http://pkg.robjhyndman.com/forecast`.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*. 22.4, pp. 679 –688.

Kaur, T., Kumar, S., and Segal, R. (2016). Application of Artificial Neural Network for short term wind speed forecasting. *2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE)*, pp. 1 –5.

Keles, D., Scelle, J., Paraschiv, F., and Fichtner, W. (2016). Extended forecast methods for day-ahead electricity spot prices applying Artificial Neural Networks. *Applied Energy*. 162, pp. 218 –230.

Khwaja, A., Naeem, M., Anpalagan, A., Venetsanopoulos, A., and Venkatesh, B. (2015). Improved short-term load forecasting using Bagged Neural Networks. *Electric Power Systems Research*. 125, pp. 109 –115.

Kim, K. J. (2003). Financial time series forecasting using Support Vector Machines. *Neurocomputing*. 55.1, pp. 307 –319.

Lahiri, S. and Lahiri, S. (2003). *Resampling Methods for Dependent Data*. Springer Series in Statistics. Springer. Available from: `https://books.google.co.za/books?id=e4f8sqm439UC`.

Makridakis, S., Wheelwright, C., and Hyndman, R. (1998). *Forecasting Methods and Applications*. 3rd ed. John Wiley & Sons, Inc.

Makridakis, S. and Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*. 16.4, pp. 451 –476.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (Mar. 2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*. 13.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2018). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. Available from: `https://CRAN.R-project.org/package=e1071`.

Mignani, S. and Rosa, R. (1995). The moving block bootstrap to assess the accuracy of statistical estimates in Ising model simulations. *Computer Physics Communications*. 92.2, pp. 203 –213. ISSN: 0010-4655.

Oliveira, P. J., Steffen, J. L., and Cheung, P. (2017). Parameter Estimation of Seasonal ARIMA Models for Water Demand Forecasting Using the Harmony Search Algorithm. *Procedia Engineering*. 186, pp. 177 –185.

Pai, P.-F., Lin, K.-P., Lin, C.-S., and Chang, P.-T. (2010). Time series forecasting by a seasonal support vector regression model. *Expert Systems with Applications*. 37.6, pp. 4261 –4265.

Panapakidis, I. P. and Dagoumas, A. S. (2016). Day-ahead electricity price forecasting via the application of Artificial Neural Network based models. *Applied Energy*. 172, pp. 132 –151.

Politis, D. N. and Paparoditis, E. (2001). Tapered block bootstrap. *Biometrika*. 88.4, pp. 1105–1119.

Quenouille, M. H. (1949). The Joint Distribution of Serial Correlation Coefficients. *The Annals of Mathematical Statistics*. 20.4, pp. 561–571.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. Available from: `https://www.R-project.org/`.

Ramos, P., Santos, N., and Rebelo, R. (2015). Performance of state space and ARIMA models for consumer retail sales forecasting. *Robotics and Computer-Integrated Manufacturing*. 34, pp. 151 –163.

Sapankevych, N. I. and Sankar, R. (2009). Time Series Prediction Using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine*. 4.2, pp. 24 –38.

Shao, X. (2010). The Dependent Wild Bootstrap. *Journal of the American Statistical Association*. 105.489, pp. 218–235.

Sánchez Lasheras, F., de Cos Juez, F. J., Suárez Sánchez, A., Krzemień, A., and Riesgo Fernández, P. (2015). Forecasting the COMEX copper spot price by means of Neural Networks and ARIMA models. *Resources Policy*. 45, pp. 37 –43.

Suykens, J. and Vandewalle, J. (1999). Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*. 9.3, pp. 293–300.

Taneja, K., Ahmad, S., Ahmad, K., and Attri, S. (2016). Time series analysis of aerosol optical depth over New Delhi using Box-Jenkins ARIMA modeling approach. *Atmospheric Pollution Research*. 7.4, pp. 585 –596.

Van den Bossche, F, Wets, G, and Brijs, T (2004). *A Regression Model with ARIMA errors to Investigate the Frequency and Severity of Road Traffic Accidents*. Available from: `http://hdl.handle.net/1942/4543`.

Vapnik, V. (1998). *Statistical Learning Theory*. New York: Wiley.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Weatherford, L., Gentry, T, and Wilamowski, B. (Jan. 2003). Neural network forecasting for airlines: A comparative analysis. *Journal of Revenue and Pricing Management*. 1, pp. 319–331.

Widowati, Putro, S. P., Koshio, S., and Oktaferdian, V. (2016). Implementation of ARIMA Model to Asses Seasonal Variability Macrobenthic Assemblages. *Aquatic Procedia*. 7, pp. 277 –284.

Yuan, C., Liu, S., and Fang, Z. (2016). Comparison of China's primary energy consumption forecasting by using ARIMA (the Autoregressive Integrated Moving Average) model and GM(1,1) model. *Energy*. 100, pp. 384 –390.

Yukun, B., Tao, X., and Zhongyi, H. (2012). Forecasting Air Passenger Traffic by Support Vector Machines with Ensemble Empirical Mode Decomposition and Slope-Based Method. *Discrete Dynamics in Nature & Society*, pp. 1 –12.

Zhang, P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*. 160, pp. 501–514.

Appendix A

R Code Used

A.1 Transforming the Air Passengers Time Series

```
#Loading all the necessary packages
library(fpp2)
library(Metrics)
library(e1071)
library(nnfor)
library(xts)
#Load the Dataset from local directory
my_data_path<-"C:/..."
passenger_data<-read.csv(my_data_path,header=T)

# Create a ts
passengers_ts<-ts(passenger_data$Overall,start=c(2012,4),frequency=12)

#Split the time series in training and testData
train_passengers_ts<-subset(passengers_ts,
                            end=length(passengers_ts)-12)
test_passengers_ts<-subset(passengers_ts,
```

```
                          start=length(passengers_ts)-11)

# Extraxt the components of air passengers time series
plot(passengers_ts,xlab="Month",ylab="Number of Passengers",
     main=" Monthly Number of Air Passengers in South Africa")
plot(decompose(passengers_ts),main="Decomposed Passengers series")
```

A.2 ARIMA Model Code

```
# check if transformation is necessary
(lambda<-BoxCox.lambda(train_passengers_ts))



#fit ARIMA Model and assess the fit
fit<-auto.arima(train_passengers_ts,stepwise=FALSE,approximation = FALSE)
forecast::accuracy(fit)
forecast::accuracy(fit$fitted,fit$x)

autoplot(train_passengers_ts)+autolayer(fit$fitted)

checkresiduals(fit) # Analyse Residuals



# produce forecasts and produce plots
fit%>%  forecast(h=12) %>%
  autoplot() + autolayer(test_passengers_ts)

test_predicted<-fit %>% forecast(12)
plot(passengers_ts,ylab="Passengers")
lines(test_predicted$mean,col="red")
```

```
legend("topleft",inset=0.05,
       title="Series Type",c("Original","Forecast"),
       lty=c(1,1),col=c("black","red"))
forecast_ARIMA<-test_predicted$mean
```

A.3 ANN Model Code

```
#fit ANN Model and assess the fit
set.seed(1234) # For reproducibility
fit<-nnetar(train_passengers_ts,p=5,repeats = 1000)
summary(fit)
forecast::accuracy(fit$fitted,fit$x)
mase(subset(train_passengers_ts,
            start=length(train_passengers_ts)-69),
     subset(fit$fitted,
            start=length(fit$fitted)-69))
autoplot(train_passengers_ts)+autolayer(fit$fitted)
forecast::checkresiduals(fit)

# produce forecasts and produce plots
test_predicted<-fit %>% forecast(12)
plot(passengers_ts,ylab="Passengers")
lines(test_predicted$mean,col="red")
legend("topleft",inset=0.05,
       title="Series Type",c("Original","Forecast"),
       lty=c(1,1),col=c("black","red"))
forecast_ANN<-test_predicted$mean
```

A.4 SVM Model Code

```
# prepare sample data in the form of
#data frame with cols of timesteps (x) and values (y)
#data(train_passenger_international_ts)

monthly_data <- unclass(train_passengers_ts)
months <- 1:length(train_passengers_ts)
DF <- data.frame(months,monthly_data)
colnames(DF)<-c("x","y")

set.seed(1234) # For reproducibility

#fit ANN Model and assess the fit using a grid of parameters
tuneResult <- tune.svm(y~x, data = DF,
                       gamma =c(0,001,0.01,0.08,0.2,.4,.6,.8,1,10),
                       cost =c(1,10,500,1000,1500,2000,2500),
                       kernel="radial",
                       type="eps-regression")

tunedModel <- tuneResult$best.model #best model based on the grid search

#specify timesteps for forecast, eg for all series + 12 months ahead
nd <- c(seq(1,94))

#compute forecast for all the  months
prognoza <- predict(tunedModel, newdata=data.frame(x=nd))

#Fit and evaluate the fit
forecasts_DF<-data.frame(prognoza)
```

```
colnames(forecasts_DF)<-c("Predicted")
SVM_predicted<-ts(forecasts_DF$Predicted,start=c(2012,4),
                  end=c(2019,1),freq=12)
forecast::accuracy (SVM_predicted,train_passengers_ts)
mase(train_passengers_ts,SVM_predicted,1)
autoplot(train_passengers_ts)+autolayer(SVM_predicted)
checkresiduals(tunedModel)

# produce forecasts and produce plots
SVM_predicted<-ts(forecasts_DF$Predicted,
                  start=c(2012,4),end=c(2020,1),freq=12)
test_predicted<-subset(SVM_predicted,start=length(SVM_predicted)-11)
forecast::accuracy (test_predicted,test_passengers_ts)
mase(test_passengers_ts,SVM_predicted,1)

plot(passengers_ts,ylab="Passengers")
lines(test_predicted,col="red")
lines(New_forecast,col="blue")
legend("topleft",inset=0.05,
       title="Series Type",c("Original","Forecast SVM","Forecast Seasonal SVM"),
       lty=c(1,1,1),col=c("black","red","blue"))
forecast_original_SVM<-test_predicted
forecast_Seasonal_SVM<-New_forecast
```

A.5 Seasonal SVM Model Code

```
################## Extract Seasonality by STL Decomposition##############
Season<-mstl(train_passengers_ts,s.window=13,t.window=50)[,3]
des_train_passengers_ts<-seasadj(mstl(train_passengers_ts,
```

```
                    s.window=13,t.window=25))# deseasonalise the time series
monthly_data <- unclass(des_train_passengers_ts)
months <- 1:length(des_train_passengers_ts)
DF <- data.frame(months,monthly_data)
colnames(DF)<-c("x","y")

set.seed(1234)

tuneResult <- tune.svm(y~x, data = DF, gamma =c(0.01,0.08,0.1,0.2),
                       cost =c(0.05,1,5,6),epsilon=c(0.008,0.01,0.1,1),
                       kernel="radial",
                       type="eps-regression")
tunedModel <- tuneResult$best.model

#specify timesteps for forecast, eg for all series + 12 months ahead
nd <- c(seq(1,94))

#compute forecast for all the  months
prognoza <- predict(tunedModel, newdata=data.frame(x=nd))

#Fit and evaluate the fit
forecasts_DF<-data.frame(prognoza)
colnames(forecasts_DF)<-c("Predicted")
SVM_predicted<-ts(forecasts_DF$Predicted,start=c(2012,4),end=c(2019,1),freq=12)
fitted_Plus_Season<-SVM_predicted+Season
Fitted_SVM<-SVM_predicted
autoplot(train_passengers_ts)+
autolayer(fitted_Plus_Season)
errors<-des_train_passengers_ts-New_fitted
```

```
forecast::accuracy (New_fitted,des_train_passengers_ts)
mase(train_passengers_ts,SVM_predicted,1)
checkresiduals(tunedModel)

# produce forecasts and produce plots
SVM_predicted<-ts(forecasts_DF$Predicted,start=c(2012,4),end=c(2020,1),freq=12)
test_forecast<-subset(SVM_predicted,start=length(SVM_predicted)-11)
Season_forecast<-ts(subset(Season,
                           start=length(Season)-11),start=c(2019,2),freq=12)
New_forecast<-test_forecast+Season_forecast
forecast::accuracy (New_forecast,test_passengers_ts)
mase(test_passengers_ts,New_forecast,1)
Seasonal_SVM<-New_forecast
autoplot(passengers_ts)+
  autolayer(New_forecast)+
  autolayer(SVM_predicted)
```

A.6 Regression Model With ARIIMA Errors Code

```
#Fit and evaluate the fit
x=seq(1,82,1)
x_test=seq(83,94,1)
fit<-auto.arima(train_passengers_ts,xreg=x,stepwise=FALSE,approximation=FALSE)
forecast::accuracy(fit)
autoplot(train_passengers_ts)+autolayer(fit$fitted)
forecast::accuracy(fit$fitted,fit$x)
mase(train_passengers_ts,fit$fitted)
checkresiduals(fit)

# produce forecasts and plots
```

```
forecast<- fit %>% forecast(12,xreg=x_test) %>%
  forecast::accuracy(test_passengers_ts)

fit %>%  forecast(h=12,xreg=x_test) %>%
  autoplot() + autolayer(test_passengers_ts)+
  guides(colour=guide_legend(title="Test"))+ylab("Cases")

test_predicted<-fit %>% forecast(h=12,xreg=x_test)
plot(passengers_ts,ylab="Passengers")
lines(test_predicted$mean,col="red")
legend("topleft",inset=0.05,
       title="Series Type",c("Original","Forecast"),
       lty=c(1,1),col=c("black","red"))
```

A.7 Bagging Code

```
bootstraps<-10
sim <- bld.mbb.bootstrap(train_passengers_ts, bootstraps) %>%
  as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))

fc <- purrr::map(as.list(sim),
                 function(p){forecast(auto.arima(p,approximation = FALSE,
                stepwise = FALSE))[["mean"]]}) %>%
                as.data.frame() %>%
                ts(frequency=12, start=c(2019,2))

v= c()
means<-c()
```

```
N<-1
while(N<=12)
{n<-1
while(n<=bootstraps)
{v[n]<-fc[[N,n]]
n<-n+1}
means[N]<-mean(v)
N<-N+1}
Bagged_forecast_ARIMA<-ts(means,start=c(2019,2),freq=12)
forecast::accuracy(Bagged_forecast_ARIMA,test_passengers_ts)
autoplot(train_passengers_ts) +
  autolayer(sim, colour=TRUE) +
  autolayer(train_passengers_ts, colour=FALSE) +
  guides(colour="none")

autolayer(Bagged_forecast, colour=TRUE) +
  autolayer(train_passengers_ts, colour=FALSE) +
  autolayer(test_passengers_ts, colour=FALSE) +
  ylab("Bootstrapped series") +
  guides(colour="none")

####################BAGGED NNETAR#########################################
bootstraps<-10
sim <- bld.mbb.bootstrap(train_passengers_ts, bootstraps) %>%
  as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))
fc <- purrr::map(as.list(sim),
                 function(t){forecast(nnetar(t,p=5,repeats = 1000))[["mean"]]}) %>%
  as.data.frame() %>%
```

```
  ts(frequency=12, start=c(2019,2))


v= c()
means<-c()
N<-1
while(N<=12)
{n<-1
while(n<=bootstraps)
{v[n]<-fc[[N,n]]
n<-n+1}
means[N]<-mean(v)
N<-N+1}
Bagged_forecast_ANN<-ts(means,start=c(2019,2),freq=12)
forecast::accuracy(Bagged_forecast_ANN,test_passengers_ts)
autoplot(train_passengers_ts) +
  autolayer(sim, colour=TRUE) +
  autolayer(Bagged_forecast, colour=TRUE) +
  autolayer(train_passengers_ts, colour=FALSE) +
  autolayer(test_passengers_ts, colour=FALSE) +
  ylab("Bootstrapped series") +
  guides(colour="none")

#######################BAGGED SVM ##########################################
bootstraps<-10
sim <- bld.mbb.bootstrap(train_passengers_ts, bootstraps) %>%
  as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))

SVM_Bagged<-function(p){
```

```
monthly_data <- unclass(p)

months <- 1:length(p)

DF <- data.frame(months,monthly_data)

colnames(DF)<-c("x","y")

set.seed(1234)

tuneResult <- tune.svm(y~x, data = DF,

                       gamma =c(0.01,0.08,0.1,0.2),

                       cost =c(0.05,1,5,6),

                       epsilon=c(0.008,0.01,0.1,1),

                       kernel="radial",

                       type="eps-regression")


tunedModel <- tuneResult$best.model


#specify timesteps for forecast, eg for all series + 12 months ahead

nd <- c(seq(1,94))


#compute forecast for all the  months

prognoza <- predict(tunedModel, newdata=data.frame(x=nd))


#Fit and evaluate the fit

forecasts_DF<-data.frame(prognoza)

colnames(forecasts_DF)<-c("Predicted")

SVM_predicted<-ts(forecasts_DF$Predicted,start=c(2012,4),end=c(2020,1),freq=12)

#test_forecast<-subset(SVM_predicted,start=length(SVM_predicted)-11)


return<- SVM_predicted

}
```

```
fc<-purrr::map(as.list(sim),SVM_Bagged)%>% as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))
tail(fc)
fc[[94,1]]
v= c()
means<-c()
N<-83
while(N<=94)
{n<-1
while(n<=bootstraps)
{v[n]<-fc[[N,n]]
n<-n+1}
means[N]<-mean(v)
N<-N+1}

Bagged_forecast_SVM<-ts(na.trim(means),start=c(2019,2),freq=12)
forecast::accuracy(Bagged_forecast_SVM,test_passengers_ts)
autoplot(test_passengers_ts) +
  autolayer(Bagged_forecast, colour=TRUE) +
  autolayer(Bagged_forecast_Season_SVM, colour=FALSE) +
  autolayer(test_passengers_ts, colour=FALSE) +
  ylab("Bootstrapped series") +
  guides(colour="none")

################## BAGGED Seasonal SVM ##########################
bootstraps<-10
sim <- bld.mbb.bootstrap(train_passengers_ts, bootstraps) %>%
  as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))
```

```
SVM_Bagged<-function(p){
  Season<-mstl(p)[,3]
  des_train_passengers_ts<-seasadj(mstl(p))# deseasonalise the time series
  monthly_data <- unclass(des_train_passengers_ts)
  months <- 1:length(des_train_passengers_ts)

  DF <- data.frame(months,monthly_data)
  colnames(DF)<-c("x","y")

  set.seed(1234)
  tuneResult <- tune.svm(y~x, data = DF,
                         gamma =c(0.01,0.08,0.1,0.2),
                         cost =c(0.05,1,5,6),
                         epsilon=c(0.008,0.01,0.1,1),
                         kernel="radial",
                         type="eps-regression")

  tunedModel <- tuneResult$best.model

  #specify timesteps for forecast, eg for all series + 12 months ahead
  nd <- c(seq(1,94))
  #compute forecast for all the  months
  prognoza <- predict(tunedModel, newdata=data.frame(x=nd))

  #Fit and evaluate the fit
  forecasts_DF<-data.frame(prognoza)
  colnames(forecasts_DF)<-c("Predicted")
  SVM_predicted<-ts(forecasts_DF$Predicted,start=c(2012,4),end=c(2020,1),freq=12)
```

```
  #test_forecast<-subset(SVM_predicted,start=length(SVM_predicted)-11)
  Season_forecast<-ts(subset(Season,
                             start=length(Season)-11),start=c(2019,2),freq=12)
  New_forecast<-SVM_predicted+ts(c(Season,Season_forecast),start=c(2012,4),freq=12)
  return<-New_forecast
}
#modell<-SVM_Bagged(unclass(sim[,3]))
#MM<-purrr::map(as.list(sim),SVM_Bagged)
fc<-purrr::map(as.list(sim),SVM_Bagged)%>% as.data.frame() %>%
  ts(frequency=12, start=c(2012,4))

v= c()
means<-c()
N<-83
while(N<=94)
{n<-1
while(n<=bootstraps)
{v[n]<-fc[[N,n]]
n<-n+1}
means[N]<-mean(v)
N<-N+1}
Bagged_forecast_season_SVM<-ts(na.trim(means),start=c(2019,2),freq=12)
forecast::accuracy(Bagged_forecast_season_SVM,test_passengers_ts)

autoplot(test_passengers_ts) +
  autolayer(Bagged_forecast_season_SVM, colour=TRUE) +
  autolayer(Seasonal_SVM, colour=TRUE) +
  guides(colour="none")
```

```
####BAGGED regression model with ARIMA errors#####################

bootstraps<-10

sim <- bld.mbb.bootstrap(train_passengers_ts, bootstraps) %>%

  as.data.frame() %>%

  ts(frequency=12, start=c(2012,4))


fc <- purrr::map(as.list(sim),

                 function(p){forecast(auto.arima(p,approximation = FALSE,

                                                 stepwise = FALSE))[["mean"]]}) %>%

  as.data.frame() %>%

  ts(frequency=12, start=c(2019,2))


v= c()

means<-c()

N<-1

while(N<=12)

{n<-1

while(n<=bootstraps)

{v[n]<-fc[[N,n]]

n<-n+1}

means[N]<-mean(v)

N<-N+1}

Bagged_forecast_ARIMA<-ts(means,start=c(2019,2),freq=12)

forecast::accuracy(Bagged_forecast_ARIMA,test_passengers_ts)

autoplot(train_passengers_ts) +

  autolayer(sim, colour=TRUE) +

  autolayer(train_passengers_ts, colour=FALSE) +

  guides(colour="none")
```

```
autolayer(Bagged_forecast, colour=TRUE) +
  autolayer(train_passengers_ts, colour=FALSE) +
  autolayer(test_passengers_ts, colour=FALSE) +
  ylab("Bootstrapped series") +
  guides(colour="none")
```

Abstract

More time series forecasting methods were researched and made available in recent years. This is mainly due to the emergence of machine learning methods which also found applicability in time series forecasting. The emergence of a variety of methods and their variants presents a challenge when choosing appropriate forecasting methods.

This study explored the performance of four advanced forecasting methods: auto-regressive integrated moving averages (ARIMA); artificial neural networks (ANN); support vector machines (SVM); and regression models with ARIMA errors. To improve their performance, bagging was also applied. The performance of the different methods was illustrated using South African air passenger data collected for planning purposes by the Airports Company South Africa (ACSA). The dissertation discussed the different forecasting methods at length. Characteristics such as strengths and weaknesses and the applicability of the methods were explored. Some of the most popular forecast accuracy measures were discussed in order to understand how they could be used in the performance evaluation of the methods.

It was found that the regression model with ARIMA errors outperformed all the other methods, followed by the ARIMA model. These findings are in line with the general findings in the literature. The ANN method is prone to overfitting and this was evident from the results of the training and the test data sets. The bagged models showed mixed results with marginal improvement on some of the methods for some performance measures.

It could be concluded that the traditional statistical forecasting methods (ARIMA and the regression model with ARIMA errors) performed better than the machine learning methods (ANN and SVM) on this data set, based on the measures of accuracy used. This calls for more research regarding the applicability of the machine learning methods to time series forecasting, which will assist in understanding and improving their performance against the traditional statistical methods.

Opsomming

Die afgelope tyd is verskeie tydreeksvooruitskattingsmetodes ondersoek as gevolg van die ontwikkeling van masjienleermetodes met toepassings in die vooruitskatting van tydreekse. Die nuwe metodes en hulle variante laat ’n groot keuse tussen vooruitskattingsmetodes.

Hierdie studie ondersoek die werkverrigting van vier gevorderde vooruitskattingsmetodes: outoregressiewe, geïntegreerde bewegende gemiddeldes (ARIMA), kunsmatige neurale netwerke (ANN), steunvektormasjiene (SVM) en regressiemodelle met ARIMA-foute. Skoenlussaamvoeging is gebruik om die prestasie van die metodes te verbeter. Die prestasie van die vier metodes is vergelyk deur hulle toe te pas op Suid-Afrikaanse lugpassasiersdata wat deur die Suid-Afrikaanse Lughawensmaatskappy (ACSA) vir beplanning ingesamel is. Hierdie verhandeling beskryf die verskillende vooruitskattingsmetodes omvattend. Sowel die positiewe as die negatiewe eienskappe en die toepasbaarheid van die metodes is uitgelig. Bekende prestasiemaatstawwe is ondersoek om die prestasie van die metodes te evalueer.

Die regressiemodel met ARIMA-foute en die ARIMA-model het die beste van die vier metodes gevaar. Hierdie bevinding strook met dié in die literatuur. Dat die ANN-metode na oormatige passing neig, is deur die resultate van die opleidings- en toetsdatastelle bevestig. Die skoenlussamevoegingsmodelle het gemengde resultate opgelewer en in sommige prestasiemaatstawwe vir party metodes marginaal verbeter.

Op grond van die waardes van die prestasiemaatstawwe wat in hierdie studie gebruik is, kan die gevolgtrekking gemaak word dat die tradisionele statistiese vooruitskattingsmetodes (ARIMA en regressie met ARIMA-foute) op die gekose datastel beter as die masjienleermetodes (ANN en SVM) presteer het. Dit dui op die behoefte aan verdere navorsing oor die toepaslikheid van tydreeksvooruitskatting met masjienleermetodes om hul prestasie vergeleke met dié van die tradisionele metodes te verbeter.

Setsopolwa

Go nyakišišitšwe ka ga mekgwa ye mentši ya go akanya ka ga molokoloko wa dinako le go dirwa gore e hwetšagale mo mengwageng ye e sa tšwago go feta. Se ke ka lebaka la go tšwelela ga mekgwa ya go ithuta ya go diriša metšhene yeo le yona e ilego ya dirišwa ka kakanyong ya molokolokong wa dinako. Go tšwelela ga mehutahuta ya mekgwa le go fapafapana ga yona go tšweletša tlhohlo ge go kgethwa mekgwa ya maleba ya go akanya.

Dinyakišišo tše di lekodišišitše go šoma ga mekgwa ye mene ya go akanya yeo e gatetšego pele e lego: ditekanyotshepelo tšeo di kopantšwego tša poelomorago ya maitirišo (ARIMA); dinetweke tša maitirelo tša nyurale (ANN); metšhene ya bekthara ya thekgo (SVM); le mekgwa ya poelomorago yeo e nago le diphošo tša ARIMA. Go kaonafatša go šoma ga yona, nepagalo ya go ithuta ka metšhene le yona e dirišitšwe. Go šoma ga mekgwa ye e fepafapanego go laeditšwe ka go šomiša tshedimošo ya banamedi ba difofane ba Afrika Borwa yeo e kgobokeditšwego mabakeng a dipeakanyo ke Khamphani ya Maemafofane ya Afrika Borwa (ACSA). Sengwalwanyakišišo se ahlaahlile mekgwa ya kakanyo ye e fapafapanego ka bophara. Dipharologanyi tša go swana le maatla le bofokodi le go dirišega ga mekgwa di ile tša šomišwa. Magato a mangwe ao a tumilego kudu a kakanyo ye e nepagetšego a ile a ahlaahlwa ka nepo ya go kwešiša ka fao a ka šomišwago ka gona ka tshekatshekong ya go šoma ga mekgwa ye.

Go hweditšwe gore mokgwa wa poelomorago wa go ba le diphošo tša ARIMA o phadile mekgwa ye mengwe ka moka, gwa latela mokgwa wa ARIMA. Dikutollo tše di sepelelana le dikutollo ka kakaretšo ka dingwaleng. Mokgwa wa ANN o ka fela o fetišiša gomme se se bonagetše go dipoelo tša tlhahlo le dihlopha tša teko ya tshedimošo. Mekgwa ya nepagalo ya go ithuta ka metšhene e bontšhitše dipoelo tšeo di hlakantšwego tšeo di nago le kaonafalo ye kgolo go ye mengwe mekgwa ya go ela go phethagatšwa ga mešomo.

Go ka phethwa ka gore mekgwa ya setlwaedi ya go akanya dipalopalo (ARIMA le mokgwa wa poelomorago wa go ba le diphošo tša ARIMA) e šomile bokaone go phala mekgwa ya go ithuta ka metšhene (ANN le SVM) ka mo go sehlopha se sa tshedimošo, go eya ka magato a nepagalo ya magato ao a šomišitšwego. Se se nyaka gore go dirwe dinyakišišo tše dingwe mabapi le go dirišega ga mekgwa ya go ithuta ka metšhene mabapi le go akanya molokoloko wa dinako, e lego seo se tlago thuša go kwešiša le go kaonafatša go šoma ga yona kgahlanong le mekgwa ya setlwaedi ya dipalopalo.